

Contrast Documentation

April 24 2025 EOP 3.12.2

This document contains guidance on the core, supported, and recommended way to use Contrast Security products.

Table of Contents

Welcome to Contrast	15
Access control and customization	15
Next steps	17
How Contrast works	17
How Contrast integrates with your development environment	18
Analysis techniques and data sources	18
Contrast agents	18
Agent configuration	18
Static scans	19
Protection for cloud-native applications	19
Integrations	19
Contrast walk through	19
Customize the Contrast environment	20
Step 1: Configure applications for security testing	22
Step 2: Configure applications to block attacks	24
Step 3: Fix code and retest applications	26
Hosted versus on-premises deployment	26
Benefits and drawbacks of hosted solutions	26
Benefits and drawbacks of on-premises solutions	26
Contrast feature comparison	27
Assess	28
Features	29
Customization	29
SCA	29
Features	29
Contrast data	30
Static SCA	30
Protect	30
How Protect works	31
Customization	31
Contrast Protect licensing guide	31
Scan	33
Features	33
See also	33
Serverless Application Security	33
Features	34
Benefits	34
How it works	34
Security and privacy	35
See also	36
Performance	36
See also	36
Community Edition (CE)	36
Community Edition features	36
Community Edition portal	37
Next steps	37
🔗 Contrast for developers	38
The Contrast platform	38
Code analysis during development	38
Analysis of open source libraries	38
Code analysis during runtime	39
Protection for production builds	39
Next step	39

🔗 Contrast analysis paths	39
Analysis paths during development	40
Analysis paths for open source libraries	40
Analysis path during runtime	40
Protection path for production builds	40
🔗 Code analysis	40
Next steps	41
Use CLI for open source libraries	41
🔗 Use CLI for static scanning	42
🔗 Use CLI for serverless function scanning	42
🔗 Use CLI to find vulnerabilities	43
Use GitHub app for open source libraries	44
🔗 Connect repos for open source library analysis	44
🔗 Use GitHub action for static scanning	45
Instrument applications for open source libraries	45
🔗 Instrument applications to find vulnerabilities	46
🔗 Use Contrast web interface to set up function scanning	46
🔗 Use Contrast web interface for static scanning	46
🔗 Analysis results	47
🔗 Get results from the CLI	47
🔗 Get results from IDE integration	47
🔗 Get results in SARIF files	48
🔗 Get results in the Contrast web interface	48
🔗 Monitor or block attacks	49
🔗 Instrument applications for Protect	49
🔗 View attack data in the Contrast web interface	49
CI/CD integration options	50
Agents	51
Install an agent	52
Java	52
.NET Framework	53
.NET Core	53
Node.js	54
PHP	54
Python	54
Ruby	55
Go	55
Download an agent configuration file	55
Java workflows	55
.NET Framework workflows	62
.NET Core workflows	67
Node.js workflows	74
PHP workflow	80
Python workflow	84
Ruby workflow	89
Go workflow	90
Ansible playbook	94
Install .NET agents with infrastructure as code tools	95
.NET agents with Terraform	95
.NET agents with Azure Resource Manager	98
Configure an agent	102
Steps	103

Find the agent keys	104
Order of precedence	106
Additional configuration	111
Exercise applications	117
Deployment to CI/CD pipeline	117
Deployment with manual testing	118
Deployment with web application test tools	118
Deployment with API test tools	118
Deployment with DAST tools	118
Deployment with open-source crawlers	119
Deployment with manual penetration testing	119
Deployment with Burp Suite-based penetration testing	119
User curl commands with Assess data	120
Java (Kotlin, Scala)	120
Supported technologies	120
Java vulnerability deduplication	122
Install	123
Configure	149
Java agent telemetry	211
.NET Framework	212
Supported technologies	212
System requirements	213
Install	214
Configure	228
Use with Azure	262
Azure Service Fabric	263
Profiler chaining	264
.NET Agent Explorer	266
Application pools	268
Telemetry	269
.NET Core	270
Supported technologies	271
System requirements	272
Install	273
Configure	290
.NET Agent Explorer	322
Profiler chaining	324
Telemetry	326
Azure functions	327
Node.js	329
Contrast service	330
Supported technologies	330
System requirements	333
Install	334
Configure	352
Reduce container startup time	387
Use the Node.js agent with ESM	389
Transpilers, compilers and source maps	390
Telemetry	391
PHP	391
Supported technologies	391
System requirements	392
Install	392
Configure	397
Python	409

Supported technologies	410
Install	411
Configure	413
Telemetry	467
Contrast Runner	467
Ruby	468
Supported technologies	469
System requirements	470
Install	471
Configure	473
Telemetry	529
Go	530
Supported technologies	530
Install the Go agent	531
Configure the Go agent	534
Contrast service	549
Install	549
Configure	550
Install	551
Configure	552
Agent Operator (Kubernetes operator)	552
Security policies	552
Custom registries	553
Pod restart	553
Next steps	553
See also	553
Supported technologies	553
Networking requirements	554
Install	554
Uninstall	570
.NET Core chaining support	571
Telemetry	572
Pod restart	572
Agent performance	573
See also	573
Performance with Protect	573
Incident management with ADR	577
ADR Benefits	577
How ADR works	577
Requirements for ADR	577
See also	577
Vulnerability assessment with AVM	578
Benefits	578
Features	578
Requirements	578
Attack events (hosted customers)	578
Event data retention	578
Exclusion of PROBED event data	579
Tasks	579
View attack events (hosted customers)	579
Manage attack events (hosted customers)	583
Attacks (on-premises customers)	584
Event data retention	585
Exclusion of PROBED event data	585
Tasks	585

View attacks (on-premises)	585
Monitor attacks (on-premises customers)	588
Manage attacks (on-premises)	589
Add tags to attacks (on-premises customers)	591
Use Contrast	593
Contrast supported languages	594
Additional technologies for Contrast Scan	596
Contrast dashboard	596
Dashboard details	596
User settings	596
Log in	597
Change password	597
Multi-factor verification	597
Manage profile	598
View API keys	598
Manage notifications	599
View permissions	599
Projects	600
See also	600
View projects	600
Export project details	602
Connect accounts	602
Applications	602
View	603
Contrast Security Observability	606
Edit application settings	611
Tag	612
Merge	613
Archive	614
Reset	615
Delete	616
Session metadata filters	616
Route coverage	619
Flow maps	631
Scans	632
Scan feature comparison	633
Scan tasks	633
See also	634
Scan release notes	634
Scan process	656
Contrast Scan supported languages and technologies	657
Scan package preparation	659
View scan projects	661
Add Scan project tags	662
Create metadata for scan projects	663
Configure dynamic scoring for Contrast Scan	664
Create a scan project	665
Delete scan projects	667
Start a scan	668
Cancel a scan	669
Monitor scans	669
SAST reports	670
Contrast Scan local engine	672
Analyze results	887
View scan policies	905

Change scan settings	905
Archive scan projects	906
Unarchive scan projects	906
Integrate scans with build pipelines	907
Servers	911
Server settings	911
Settings in a configuration file	911
Agent configuration instructions	911
Contrast options	912
View servers	912
Configure server settings	915
Application sampling	916
Automatic diagnostic collection	916
Output to syslog	918
Libraries	922
See also	922
SCA release notes	923
Contrast SCA supported languages	923
View libraries	924
Discover or delete	928
Tags	929
Send	930
Runtime library usage	930
Export	932
Open-source licenses	933
View dependency trees	934
Library scoring guide	934
Libraries (Repo)	935
CVE search	938
Serverless	938
See also	938
Release notes	938
Contrast Serverless supported languages	964
Contrast Serverless supported platforms	965
Multi-region support	965
Inventory	966
Scan types and monitoring	967
Get started with Contrast Serverless for AWS	968
Get started with Contrast Serverless for Azure	974
Scan functions on demand	975
View results	977
Change inventory criteria	982
Change serverless scan settings	983
View function and service relationships	984
Contextual risk scores	988
Upgrade Contrast Serverless	990
Block accounts	993
Offboard	994
Uninstall	994
Contrast CLI	994
Before you start	994
About Contrast CLI	994
Contrast CLI supported languages and package managers	995
Install Contrast CLI	995
Authenticate	996

Analyze	996
Contrast CLI commands	1002
Legacy Contrast CLI	1014
Vulnerabilities	1021
Contrast AI guidance	1022
View for organization	1022
View application vulnerabilities	1024
View vulnerability rates	1026
Add and delete vulnerabilities	1027
Group vulnerabilities by sink	1028
Merge vulnerabilities	1028
Tag	1028
Track	1029
Map ADR rules to Assess findings	1031
Events	1032
Fix	1033
Export	1033
Find CWEs associated with CVEs	1034
Status	1035
Edit severity	1039
Contrast Security GitHub App	1039
How it works	1039
Next steps	1040
Contrast SCA supported languages	1040
Install and authorize	1041
GitHub repository connections	1042
Troubleshoot	1043
Reports	1043
Attestation reports	1043
DISA STIG Viewer checklists	1045
Software bill of materials	1046
Vulnerability trend reports	1047
Organization statistics	1049
Remediation summary	1049
Integrations	1051
ADR integrations	1051
Platform integrations	1052
Integrations	1052
Integration release notes	1060
.....	1060
Agile Central	1061
Manage credentials	1062
AWS Security Hub	1063
Before you begin	1063
Configure	1063
Set up applications in Contrast Assess	1064
Retry mechanism	1064
Amazon Security Lake	1064
Before you begin	1064
Create a Custom Source in AWS	1064
Connect to Amazon Security Lake	1064
Set up applications in Contrast Assess	1065
Retry mechanism	1065
Azure Boards	1065
Azure Boards requirements	1065

See also	1066
Connect to Azure Boards	1066
Manage Azure Boards credentials	1066
Auto create tickets	1067
Two-way integration	1067
Exclude sensitive information with Azure Boards	1068
Personal access tokens	1069
Azure Pipelines	1069
Install and configure	1069
Configure a task	1070
Add a release gate	1071
Find logs for Azure Pipelines	1071
Azure Service Fabric	1072
Bamboo	1073
Install	1073
Configure thresholds	1074
Eclipse	1075
Generic webhooks	1076
Generic webhooks	1076
Variables	1077
Events and variables	1080
GitHub	1080
See also	1081
GitHub and GitHub Advanced Security	1081
Integration example: Assess and GitHub	1083
GitHub Actions	1084
Gradle	1084
Sample application	1084
Use the plugin	1085
IntelliJ plugin	1085
Before you begin	1086
Install Java Development Kit (JDK)	1086
Configure the Heap Size	1088
Install the Contrast IntelliJ plugin	1088
Jenkins	1089
Install and use Jenkins plugin	1089
See also	1089
Define a connection	1090
System security controls	1091
Job level security controls	1091
Pipeline security controls	1092
Jenkins security controls	1093
Define a job outcome policy	1093
Run a build	1097
Jira	1097
See also	1097
Connect to Jira	1097
Configure Jira for Assess	1098
Configure Jira for Serverless	1100
Manage credentials	1101
Two-way integration	1102
Maven	1103
See also	1103
Microsoft Teams	1104
Before you begin	1104

Create an instant cloud flow in Microsoft Teams	1104
Configure the Instant cloud flow trigger	1104
Configure the Contrast integration	1104
PagerDuty	1105
Solutions Business Manager	1105
ServiceNow	1106
Connect to ServiceNow	1106
Slack	1107
Splunk	1107
Before you begin	1107
Step1: Install Contrast Security App	1108
Step 2: Set up a syslog receiver in Splunk	1108
Step 3: Set up Contrast agents	1108
Step 4: View Contrast data in Splunk	1109
See also	1109
Splunk with ADR	1110
How it works	1110
Before you begin	1110
Step1: Install the Contrast ADR app in Splunk	1110
Step 2: Set up Splunk CIM	1111
Step 3: Set up an HTTP Event Collector input in Splunk	1111
Step 4: Configure Contrast Security ADR to send Attack Events to Splunk	1111
Step 5: Set up macros to search for events	1112
Step 6: Set up CIM data models	1112
Step 7: Set up API details	1112
View Contrast ADR data in Splunk	1113
See also	1113
VictorOps	1113
Visual Studio	1114
Visual Studio Code	1114
Visual Studio for Mac	1116
Wiz	1116
Before you begin	1116
Turn on the Contrast integration in Wiz	1117
Connect to Wiz	1117
Administration	1118
Rules and policy	1118
Assess rules	1119
Security controls	1120
Vulnerability policy	1124
Protect rules	1130
CVE shields	1134
Virtual patches	1138
Log enhancers	1140
Application exclusions	1142
Set compliance policy	1147
IP management	1148
Library policy	1151
Sensitive data masking	1151
Notifications	1153
Organization	1153
Enable Assess	1153
Enable Protect	1155
Configure	1158
Notifications	1175

Score	1177
Vulnerability approval	1178
View audit log (hosted customers) ☺	1178
View audit log	1180
Impersonation	1183
System administration	1185
Get started	1185
Contrast installation	1186
Next steps	1186
Contrast system requirements	1186
Sizing recommendations	1187
Download Contrast with curl	1188
Download Contrast installer	1189
Install	1190
Deploy Contrast with a WAR file	1193
Distributed MySQL	1194
Distributed deployment	1197
Run Contrast	1199
Credentials	1200
Restart Contrast	1201
Uninstall	1201
Post-installation	1202
Post-installation tasks	1202
Configure Tomcat	1202
JRE	1203
Configure HTTPS	1203
Configure HTTP headers	1206
Customize MySQL	1207
Set up a proxy configuration	1207
Configure reporting storage	1209
Contrast logs	1210
Use Redis as a shared cache (on-premises)	1212
System updates and upgrades	1213
Updates and upgrades	1213
Upgrade Contrast	1213
Upgrade agents (on-premises)	1214
Update your IP address	1215
Upgrade SCA library data manually	1215
Upgrade SCA library data automatically	1216
Update Contrast license	1217
Manage administration	1218
Manage multiple organizations	1218
Add/edit an organization	1219
Users and permissions	1221
Add a user	1221
Add multiple users	1222
Designate SuperAdmins	1224
Add access group	1224
Grant Protect permissions (on-premises)	1226
Auto-add users	1227
Credentials	1229
Impersonate users	1230
Authentication	1231
Multi-factor authentication	1232
Active Directory	1233

LDAP	1237
SSO	1242
HTTPS proxy	1246
Passwords	1246
Keys	1248
System settings	1248
Steps	1249
Additional system settings	1249
General	1250
Diagnostics	1250
Licenses	1251
Score	1254
Library compliance policy	1254
Mail	1254
System maintenance	1255
Encrypted properties editor	1255
MySQL backups	1257
Manage SSL	1259
Reference	1260
Glossary	1260
Open source software attributions	1264
Roles and permissions	1264
Application roles	1264
Organization roles	1267
System roles	1269
Application scoring guide	1269
Library scoring guide	1270
Log levels	1271
Events and variables	1272
Variables	1272
Regular expressions	1274
Supported browsers	1275
Beta Terms and Conditions	1275
Privacy and data collection	1275
Preview releases	1277
Topics in this section	1277
Access control (Preview)	1277
Access control settings	1277
Methods for managing access control	1278
Naming standards and requirements	1278
Access Control APIs	1278
Actions and permissions (Preview) ☹	1278
Resource groups (Preview)	1284
Roles (Preview) ☹	1291
Users (Preview) ☹	1300
User access groups (Preview) ☹	1309
Access control troubleshooting (Preview)	1314
Report dashboard (Preview)	1318
Report dashboard details	1319
See also	1320
View report dashboard (Preview)	1320
Aggregated Vulnerability report	1320
Visual Studio Code (Preview)	1320
Before you begin	1321
Download Visual Studio Code	1321

Install Visual Studio Code	1321
Launch Visual Studio Code	1322
Install the Contrast Visual Studio Code plugin	1322

Welcome to Contrast

Contrast supports real-time application security through all phases of your software development life cycle (SDLC).

Take a walk through (page 19) an example of how you can use Contrast in your environment.

If you want to...	Contrast offers...
Analyze your applications for security vulnerabilities during the development and test (QA) phases of your SDLC: <ul style="list-style-type: none">• In the development phase: Get instant and accurate vulnerability feedback for applications and the libraries that they use. By exercising your application, you can simulate the routes in your application and, with the data from Contrast, ensure that you are checking in secure code.• In the test phase: Get assurance that applications are evaluated for security vulnerabilities as you apply manual or automated test cases or in a CI/CD pipeline.• In production: Get full visibility into attacks and defend applications from malicious exploitation in the production phase of your SDLC.	<ul style="list-style-type: none">• Agents (page 51) Supports a variety of programming languages, frameworks, and container technologies that instrument your applications with sensors.• Contrast Assess: (page 28) Uses tuneable detection rules to accurately find vulnerabilities. It provides details on how the issue was discovered, how to reproduce it, and how to fix it.• Scan: (page 33) Identifies vulnerabilities in uploaded binary packages by performing a fast and efficient static scan.• Protect (a component of ADR): (page 30) Automatically identifies attacks and either monitors them or prevents them from being exploited in production. Protect discovers and blocks attacks from within the running application but can also integrate with Web Application Firewalls (WAF).
Analyze libraries that your applications use.	Contrast SCA: (page 29) Offers visibility into security risks and legal issues introduced by open-source libraries used during applications at run time. It identifies vulnerabilities in open-source libraries. It also identifies if a current library is out-of-date and should be updated.
Find vulnerabilities in your code earlier in the SDLC and get easy-to-understand guidance on how to fix them.	View vulnerability (page 1021) data that includes suggestions on how to fix (page 1033) vulnerabilities that Assess, Scan, and SCA discover.
View an architecture diagram that provides an interactive view of where data and resources are shared within your organization and beyond it.	Flow maps (page 631) provide a detailed diagram of your application, the layers of technologies within it, and the back-end systems to which it connects.
Integrate Contrast into your CI/CD pipeline.	A wide variety of integrations (page 1051) that let you to integrate Contrast actions and data into developer IDEs, build system, communication tools, and more.

Access control and customization

Contrast provides a variety of options for customizing data access, data views, and data collection from applications that you've added to Contrast. Customization helps you to enhance your views of the data that Contrast provides.

Option	Description						
Role-based access control	<p>Hosted customers:</p> <p>Resource groups (page 1284), roles (page 1291), and user access groups (page 1289) let you assign permissions and capabilities for specific users. You can associate specific applications with one or more resource groups.</p> <p>This feature provides built-in groups and roles, or you can create custom groups and roles that let you fine-tune the permissions for users</p> <p>On-premises customers:</p> <p>Access groups (page 1164) let you assign permissions and capabilities for specific users. You can assign different types of access, based on role, for each application associated with a group.</p> <p>Group strategy:</p> <p>It is useful to plan a resource group (hosted customer) or an access group (on-premises customers) strategy before you add applications to Contrast.</p> <p>If you do not specify the access or resource group in the Contrast configuration file when you first add the application to Contrast, you can only add it to a group from the Contrast web interface. If you want to add applications using a Contrast configuration file, you will need to delete the application and add it again to associate it with your access group.</p> <p>Start by creating or adding a user or application (or both) to an existing group in the Contrast web interface.</p> <p>Then, using a Contrast configuration file for each application, you can associate an application with an access group when you add the application to Contrast.</p> <pre># application: # Add the name of the application group with which this # application should be associated in the Contrast UI. # group: NEEDS_TO_BE_SET</pre>						
Custom filters	<p>Contrast provides tag options that let you create customized filters. The benefit of creating custom filters is you can view data according to your specific needs, in addition to using the default filters.</p> <p>You can create custom filters through the use of application metadata. (page 1174)</p> <p>You can also apply tags to specific application (page 612) data or vulnerability (page 1028) data in Contrast. After you tag an application or a vulnerability, you can use that tag as a filter on the Applications page or the Vulnerabilities page in the Contrast application.</p> <p>Example: Application metadata</p> <p>This example shows how to create free form, custom fields in the Contrast web interface to request application metadata:</p> <table> <tr> <td>Custom field: managersInfo</td><td>Value:"John Doe"</td></tr> <tr> <td>Custom field: businessUnit</td><td>Value:"NodeGoat Group"</td></tr> <tr> <td>Custom field: officeLocation</td><td>Value:"New York City"</td></tr> </table> <p>Example: Application tags</p> <ul style="list-style-type: none"> • Appname: The name of a specific application. • Groupname: The name of an access group. • Environment: The environment in which you are testing the application (development, QA, or production). • Server Name: The name of the server hosting the application. <p>Example: Vulnerability tags</p> <ul style="list-style-type: none"> • Build: A specific build number • Version: A specific release version 	Custom field: managersInfo	Value:"John Doe"	Custom field: businessUnit	Value:"NodeGoat Group"	Custom field: officeLocation	Value:"New York City"
Custom field: managersInfo	Value:"John Doe"						
Custom field: businessUnit	Value:"NodeGoat Group"						
Custom field: officeLocation	Value:"New York City"						

Option	Description
Custom data from applications	<p>Session metadata (page 616) lets you identify the source of vulnerabilities in your application.</p> <p>When you add the necessary property to your agent configuration file, the agent reports this information along with the rest of the standard vulnerability data to the Contrast web interface for filtering.</p> <p>If you change the values of metadata in the Contrast configuration file for the agent, you can filter the vulnerability data based on the different values. For example, if you change the values for Branch name or Version, you can filter data based on the different versions or branches.</p> <p>Example:</p> <p>In this example for a Java application, you add an entry in the line where you add your javaagent flag. In this example, you set the property <code>contrast.application.session_metadata</code> to a set of key-value pairs that identify a branch, a committer, and a repository.</p> <pre>-Dcontrast.application.session_metadata="branchName=build22,committer=Jane,repository=Contrast-Java"</pre>
Custom naming	<p>You have the option of providing customized names for applications (page 611) and servers (page 915) that host the applications.</p> <p>By default, a Contrast agent creates a name based on data it discovers in your code.</p> <p>To specify a custom name, you can use an agent configuration file when you add the application (page 52) to Contrast or set the name in the Contrast web interface after you add an application.</p>

Next steps

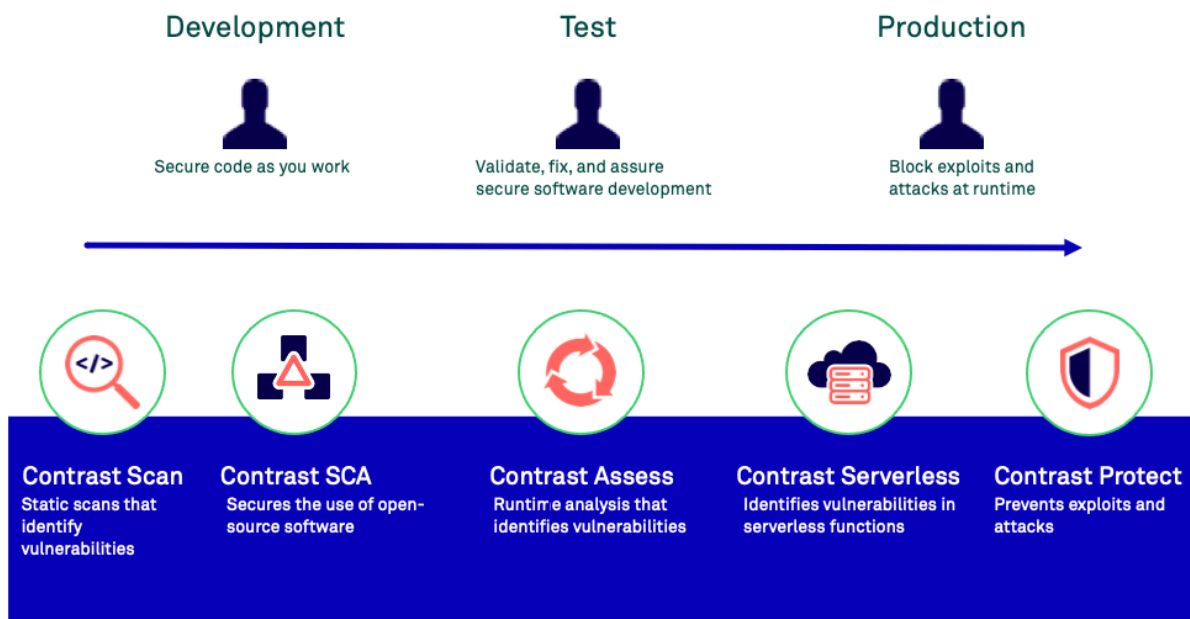
- Get an overview of [how Contrast works \(page 17\)](#)
- Try Contrast for free with [Community Edition \(CE\) \(page 36\)](#)
- [Install and configure a Contrast agent \(page 52\)](#)
- [Install Contrast \(on-premises\) \(page 1185\)](#)
- Get started with a particular [integration \(page 1051\)](#)

How Contrast works

Contrast Security provides accurate, continuous, real-time application security testing and attack blocking for your application portfolio. Contrast works within each application to secure it across the entire software development life cycle (SDLC).

Contrast transforms functional tests into security tests, so that you get security feedback every time you exercise your applications through your quality assurance function. Contrast delivers results continuously and in real time, so you are integrating security into your entire development pipeline from source code to running applications, and all points in between.

How Contrast integrates with your development environment



Analysis techniques and data sources

Contrast combines numerous data sources and a variety of analysis techniques including:

- Runtime control flow and dataflow (IAST)
- Application code or APIs (SAST)
- HTTP requests and responses
- All libraries and frameworks in the application and how they are used (SCA)
- Configuration information
- Back-end connections
- Static scans of local files (SAST)

Contrast agents

Contrast Assess and Contrast Protect use [agents \(page 51\)](#) to analyze data flow and identify vulnerabilities in fully-assembled and running applications. Contrast Assess and Contrast Protect use the same agent to analyze data flow and identify vulnerabilities in fully-assembled and running applications. You do not need one agent for Assess and another for Protect.

[Adding and configuring an agent \(page 52\)](#) inserts Contrast code in the application's existing methods across custom code and libraries. Sensors in the agents observe the locations where data enters and leaves the application (routes). This action creates real-time visibility into any data that flows through the application and allows Contrast to detect security flaws or vulnerabilities in this code path and report them to Contrast. The agents also allow Contrast to identify and block attacks.

Agent configuration

Configuring an agent consists of editing a YAML configuration file, using environment variables on a command line, or other methods native to the language and tools you are using.

When you configure an agent for an application, you specify information for the following settings:

- Agent communication with Contrast
- Agent-specific settings

- Settings for Assess and Protect rules
- Application-specific settings
 - These settings include session and application metadata that are available to you as additional information for each vulnerability reported or as a way to filter them.
- The server hosting the application and the agent:
 - Developer's local application server running in the integrated development environment (IDE)
 - Continuous integration application server that's used during the automated testing process
 - Application test server
 - Application staging server
 - Embedded server in an appliance
 - Application server running in a virtual machine
 - Remote application server running in the cloud
 - Production application server

Static scans

[Contrast Scan \(page 632\)](#) is a static application security testing (SAST) tool that makes it easy for you to find and remediate vulnerabilities during the development phase of software development lifecycle (SDLC).

To scan applications, you [upload a source code or bytecode file \(page 665\)](#). Contrast technology identifies vulnerabilities based on a set of rules that Contrast defines for you.

Protection for cloud-native applications

[Contrast Serverless Application Security \(page 33\)](#) is a next-generation application security testing solution for serverless-based applications.

Contrast Serverless Application Security uses cloud-native architecture to map all resources within your environment, while automatically validating and prioritizing the results, eliminating false-positive results and alert fatigue. It uses a ReadOnly access to your AWS account to continuously monitor the environment and collect relevant information.

Integrations

Contrast works with several different [integrations \(page 1051\)](#) to provide accurate security feedback with tools you are already using. This approach accelerates the software development process by encouraging security and development to work together effectively.

Contrast walk through

Let's take a closer look at how you can use Contrast to ensure your code is secure from critical vulnerabilities and protect your applications from attacks.



In this example, you are using the Contrast Java agent to configure applications for security testing and blocking attacks, as the application is exercised.

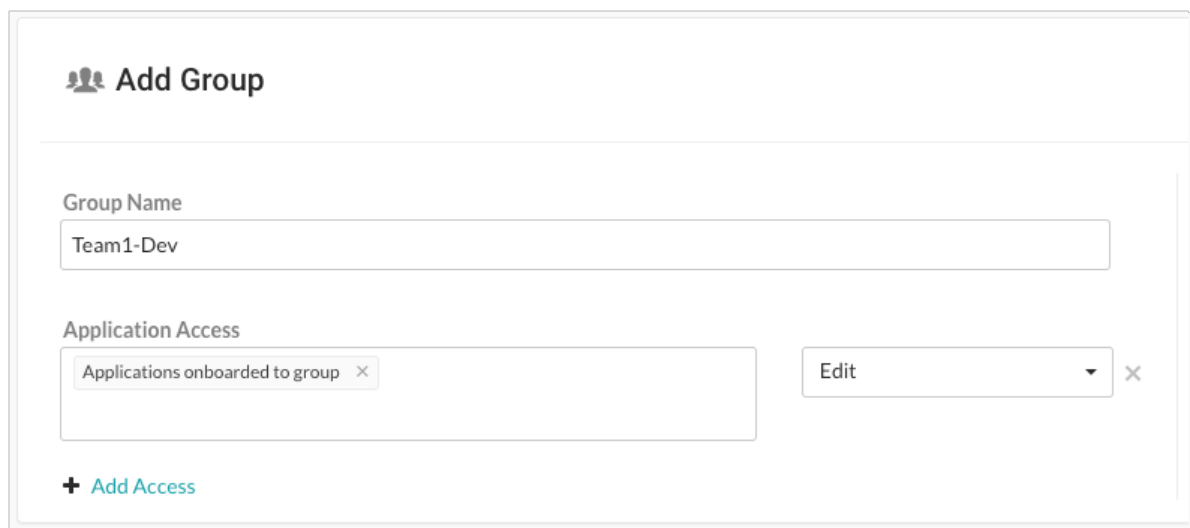
Customize the Contrast environment

Once you have access to the Contrast web interface, you think about customizing your environment so that you can find important test results easily and control access to Contrast data.

Access groups

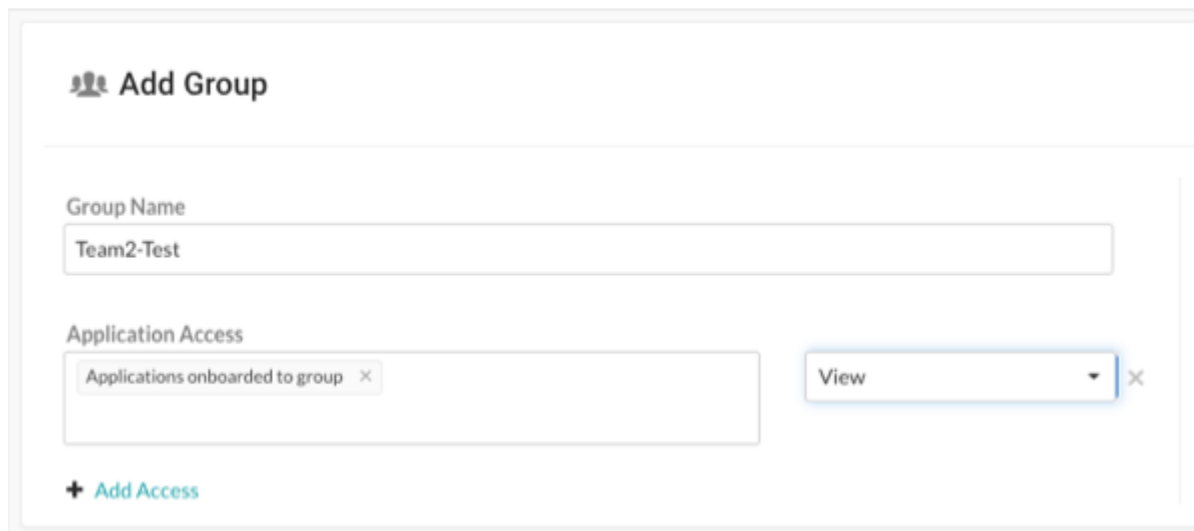
You have three teams involved with your financial applications. You create these access groups and specify them in the Contrast configuration files:

- **Team1-Dev** is for developers and has these settings:



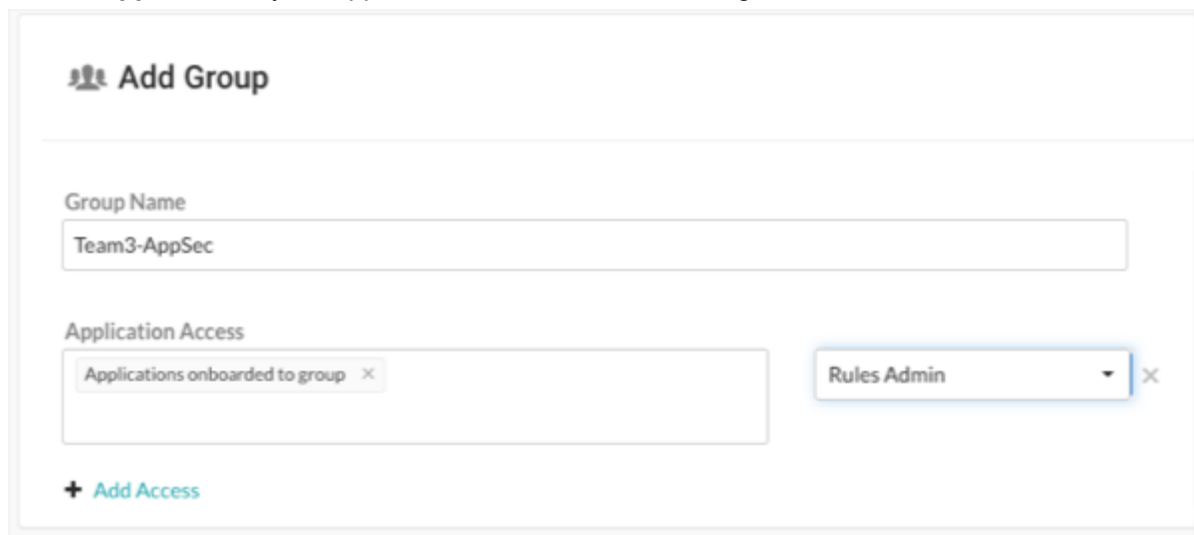
Developers can remediate findings, add tags, manage vulnerabilities, edit attributes, merge applications, add or delete applications, and create servers.

- **Team2-Test** is for your test team and has these settings:



Test staff can see scores, libraries, vulnerabilities and comments, but can't perform edits to traces to the application.

- **Team3-AppSec** is for your AppSec team and has these settings:



The AppSec team can edit rules and policies in the application, enable Protect, and manage notifications and scoring for the application.

Application and server naming

Since all teams are working on the same application, you use the same name in each Contrast configuration file. You plan to use one configuration file for the development environment and one for the test environment.

Although you are using two configuration files, since you use the same name for the application in both files, Contrast displays data as if you only have a single instance of the application.

You decide to let Contrast determine the server name.

Session metadata

You want to be able to view vulnerabilities and route information for a specific branch, committer, and repository. You define session metadata values in your Contrast configuration file to collect this information:

```
-Dcontrast.application.session_metadata="branchName=release24,committer=Jane,repository=finapp-Java"
```

**NOTE**

Most Contrast agents now automatically create a unique fingerprint that is equal to that unique build of an agent and populates session metadata values. The key used for this is called `artifactHash`. [Session metadata \(page 616\)](#) lists the supported agent versions for this feature.

Step 1: Configure applications for security testing

During development, you want to make sure developers are checking in code that's secure. During testing, you want to verify that your applications have no vulnerabilities that will allow attacks when in production.

You decide to add Contrast to your applications so that you can do the necessary security testing before releasing your products to the public. Since your applications use Java, you are going to use the Contrast Java agent.

1. You [download the agent \(page 124\)](#) from the Maven Central repository because you use Maven for your build processes. You also download a [YAML configuration file \(page 107\)](#) from the Contrast web interface.
2. For your development environment, you edit the YAML configuration file to include settings similar to these:

```
api:
  # ***** REQUIRED *****
  # Set the URL for the Contrast UI.
  url:https://mycontrast.mycompany.com:8080/Contrast/
  # ***** REQUIRED *****
  # Set the API key needed to communicate with the Contrast UI.
  api_key:A2xxxxxxxxxxxxxxxxxxxxxxxxG9N
  # ***** REQUIRED *****
  # Set the service key needed to communicate with the Contrast
  # UI. It is used to calculate the Authorization header.
  service_key:service_key:88xxxxxxxxxx5Z
  # ***** REQUIRED *****
  # Set the user name used to communicate with the Contrast
  # UI. It is used to calculate the Authorization header.
  user_name:agent_XXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX@mydevorg
.
.
.
#
=====
=====
# server
# Use the properties in this section to set
# metadata for the server hosting this agent.
#
=====
=====
# server:
# Override the reported server environment.
environment: development
```

.

.

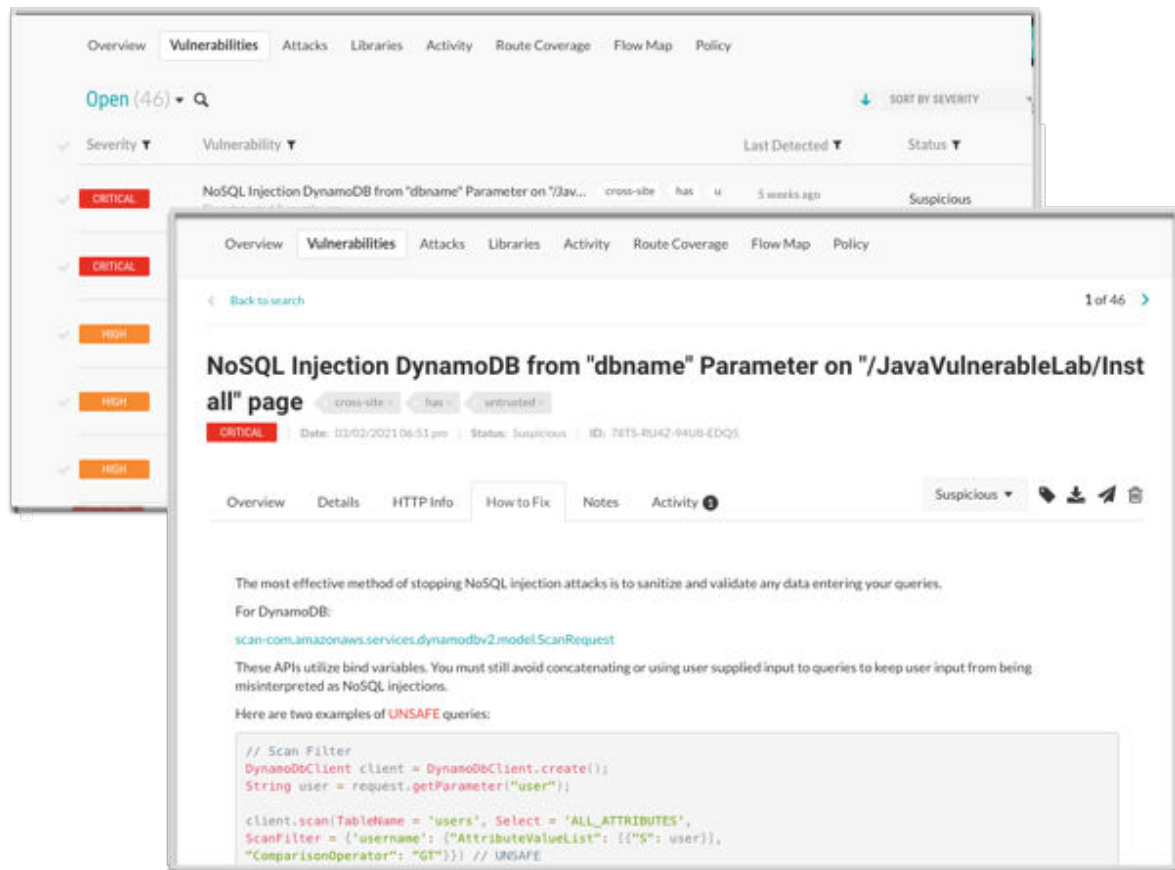
.

3. For your test environment, you edit a Contrast YAML configuration file with settings similar to these:

```
api:
  # ***** REQUIRED *****
  # Set the URL for the Contrast UI.
  url:https://mycontrast.mycompany.com:8080/Contrast/
  # ***** REQUIRED *****
  # Set the API key needed to communicate with the Contrast UI.
  api_key:A2xxxxxxxxxxxxxxxxxxxxxxxxG9N
  # ***** REQUIRED *****
  # Set the service key needed to communicate with the Contrast
  # UI. It is used to calculate the Authorization header.
  service_key:service_key:88xxxxxxxxxx5Z
  # ***** REQUIRED *****
  # Set the user name used to communicate with the Contrast
  # UI. It is used to calculate the Authorization header.
  user_name:agent_XXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX@mydevorg
.
.
.

#
=====
=====
# server
# Use the properties in this section to set
# metadata for the server hosting this agent.
#
=====
=====
# server:
# Override the reported server environment.
  environment: QA
.
.
.
```

4. Next, you start your application and run functional tests to exercise all the routes and data endpoints that the application and business logic expose.
5. Using the Contrast web interface, you first check the Application page to make sure that Contrast recognizes your application. Then, you check for vulnerabilities and how to fix guidelines to determine what actions to take to secure your code.



- After initial tests, you decide to use the [Maven plugin \(page 1103\)](#) to integrate Contrast in to your CI/CD process. You configure the integration so that builds fail if Contrast discovers vulnerabilities with a Critical or High status.

Step 2: Configure applications to block attacks

Although you've been using Contrast during your development and test phases, you also want make sure that your users are not subject to malicious activity when they use your product. You decide to add Contrast to the applications that are in production to protect your application, users, and data.

First, you make sure that you have a Protect license and that Protect is enabled for your organization.

Similar to how you installed and configured an agent for your application in development and test, you need to configure a new configuration file for the production environment that enables Contrast Protect. After you create the new configuration file, you run the application and verify that the Contrast web interface displays your application for a production environment.

```
api:
# ***** REQUIRED *****
# Set the URL for the Contrast UI.
url:https://mycontrast.mycompany.com:8080/Contrast/
# ***** REQUIRED *****
# Set the API key needed to communicate with the Contrast UI.
api_key:A2xxxxxxxxxxxxxxxxxxxxxxxxG9N
# ***** REQUIRED *****
# Set the service key needed to communicate with the Contrast
# UI. It is used to calculate the Authorization header.
service_key:service_key:88xxxxxxxxxxxx5Z
# ***** REQUIRED *****
# Set the user name used to communicate with the Contrast
# UI. It is used to calculate the Authorization header.
```

```
user_name:agent_XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX@myprodorg
.
.
.
#
=====
==
# protect
# Use the properties in this section to override Protect features.
#
=====
==
# protect:

# Use the properties in this section to determine if the
# Protect feature should be enabled. If this property is not
# present, the decision is delegated to the Contrast UI.
# enable: true
.
.
.
#
=====
==
# server
# Use the properties in this section to set
# metadata for the server hosting this agent.
#
=====
==
# server:
# Override the reported server environment.
environment: production
.
.
.
```

Once the application is in production, you monitor the Attacks page in the Contrast web interface to see if attacks occur.

Effective	Find Attack	Set Date Range	Advanced
<input type="checkbox"/> Source IP	Status	Application	Server
<input type="checkbox"/> Rule	Start	End	Events
<input type="checkbox"/> 1	EXPLOITED	Cat-Engine-2	Cat-Server
<input type="checkbox"/> 1	EXPLOITED	Cat-Engine-2	Cat-Server
<input type="checkbox"/> 1	EXPLOITED	Cat-Engine-2	Cat-Server
<input type="checkbox"/> 1	EXPLOITED	Cat-Engine-1 Cat-Engine-2	Cat-Server
<input type="checkbox"/> 1	BLOCKED IP	Grape-on-rack	Grape-Server

Step 3: Fix code and retest applications

After you analyze the results of testing with Contrast and identified attacks, you update the code and ensure that Contrast displays the latest version of your application. You verify that the new version is free of the vulnerabilities you blocked in production and re-deploy the application.

Hosted (SaaS) versus on-premises deployment

When you consider deploying Contrast Security solutions, you have two primary options: a hosted solution (cloud installation) or an on-premises instance. Each approach has its benefits and drawbacks, influenced by cost, control, customization, security, and scalability.

Benefits and drawbacks of hosted solutions

- **Benefits**
 - **Immediate access to updates and advanced new features:** Updates are readily available without delay, promoting the latest security posture. New features are always supported for hosted solutions.
 - **Reduced IT overhead:** Contrast manages infrastructure and maintenance and thus, streamlines operations. Also, freedom from system-wide management tasks.
 - **Scalability:** Easier to scale resources as your needs increase.
 - **Cost:** Pricing for SaaS deployments are subscription-based, allowing flexibility and scalability
- **Drawbacks**
 - **Data management:** Data is stored on Contrast servers, instead of locally. However, Contrast complies with these data protection policies:
 - General Data Protection Regulation (GDPR)
 - General Data Protection Regulation-UK (UK-GDPR)
 - California Consumer Privacy Act (CCPA)
 - Protection of Personal Information (APPI)
 - System and Organizational Control Type II Audit (SOC II)

Benefits and drawbacks of on-premises solutions

- **Benefits:**
 - **Complete control:** More control over system-wide settings.
 - **Data privacy:** Data is stored locally - For deployments that require specific security compliance, sensitive data never leave your company.

• Drawbacks

- **Resource intensive:** Requires significant investment in IT, networking, and infrastructure along with coordination, planning and maintenance.
- **Delayed updates:** Updates for product enhancements are often delayed after Contrast releases while for hosted solutions receive them immediately.
- **No support for new features:** Advanced new features are often not supported for on-premises solutions. For example, Contrast Scan, static SCA. GitHub App for SCA , and Contrast Serverless are not supported for on-premises instances.

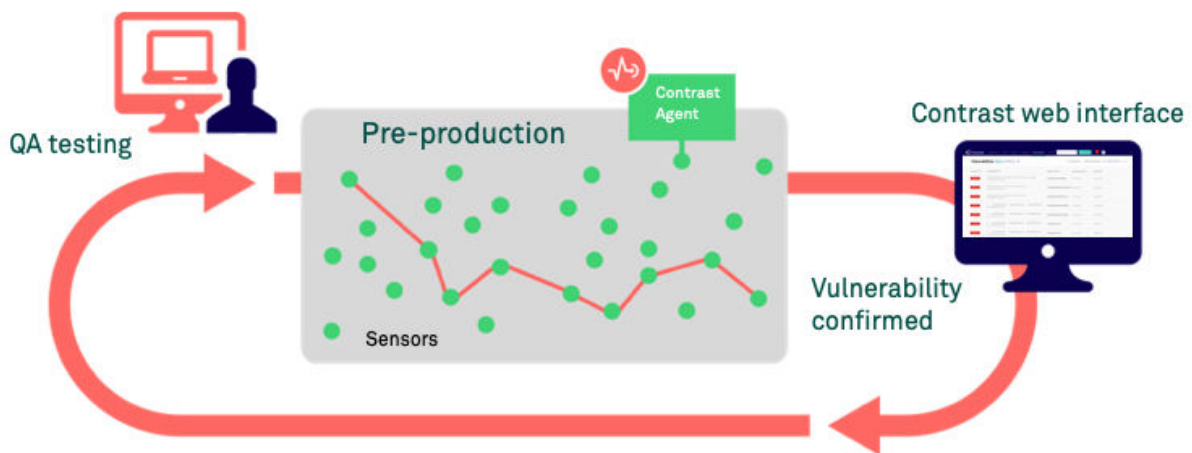
Contrast feature comparison

Feature	Hosted	On-premises
Installation and updates	Contrast installs, configures, and updates the software.	Hosted customers are responsible for installing, configuring, and updating the software.
Management at a system level	Contrast takes care of all system management tasks. With the correct permissions, a user can control a variety of configuration settings and access control entities	A SuperAdmin is responsible for all settings and configuration at a system-wide level.
Single Sign On (SSO)	Contrast Support configures authentication; however, you may be granted permissions to set up SSO for your organization.	System Administrators can configure SSO at a system-wide level.
TLS connections and certificates	For Contrast agents, Contrast uses strong TLSv1.2 connections and certificates signed by industry standard certificate authorities (CAs).	On-premises customers may need to configure Contrast agents to use enterprise CAs. They may want the agents to send client certificates in the TLS handshake.
Licenses	Hosted customers can allocate Assess and Protect licenses for their organization.	SuperAdmin or ServerAdmin role can allocate Assess and Protect licenses to a particular organization.
New integrations	Hosted customers have access to all new integrations that Contrast adds to the platform.	On-premises customers have limited access to integrations that Contrast supports.
Impersonation	Contrast support manages impersonation when needed for troubleshooting.	SuperAdmins manage impersonation when needed for troubleshooting.
Code scanning (SAST)	Hosted customers can use the Contrast scan engines from the Contrast web interface or a local scan engine. The local scan engine does not require uploading your source files to Contrast.	Not available
Software composition analysis (SCA)	Contrast Support enables this feature for the organization.	A SuperAdmin can enable SCA.
SCA repository scanning	Hosted customers can use SCA repository scanning capabilities to look for known vulnerabilities in the software components that are included in a repository.	Available except for air-gapped environments.
Static scanning of libraries	Hosted customers can automatically scan, in close to real-time, relevant static code and configuration assessments to discover new vulnerabilities.	Available except for air-gapped environments.
Organization management	Users with administrator permissions can manage their organization.	SuperAdmins and System Administrators can manage all organizations at a system-wide level.
Runtime security testing (IAST)	Available	Available
Serverless	Hosted customers can use Contrast Serverless for dynamic scanning, static scanning, graph visualization, and resource observability for AWS functions.	Not available
Software bill of materials (SBOM)	Contrast Support enables this feature for the organization. Users can generate an SBOM from the Applications tab.	A SuperAdmin can enable users to generate an SBOM from the Applications tab.
Attack protection (RASP)	Contrast Security grants permissions that let users access Protect data.	SuperAdmins can grant permissions that let all or some user roles in one or more organizations access Protect data.

Feature	Hosted	On-premises
Updated attack events user interface	Hosted customers have access to updated views of attack data that Protect provides.	Hosted customers have access to legacy attack views only.
Enhanced role-based access control (RBAC)	Hosted customers have access to an advanced access control system that lets them fine-tune roles and permissions for their organization.	Not available. On-premises customers use the legacy access control. On-premises customers can add multiple users at one time.
Contrast Security Observability	Hosted customers can access a model of an application's security architecture and behavior at runtime. This information provides a better understanding of the underlying behavior of applications for threat modeling, pen test support, and contextual information around vulnerabilities and attacks.	Not available.
Enhanced audit log	Hosted customers can access an updated and enhanced audit log view.	Not available. On-premises customers use the legacy audit log view.
Diagnostics	Contrast Support enables this option of diagnostic information is needed for troubleshooting.	A SuperAdmin, ServerAdmin or System Administrator can enable this option at a system-wide level.
Email	Users with administrator permissions can set default settings for Contrast notifications at an organization level. Individual users can adjust their own settings.	System Administrators can enable, disable, and configure Contrast to communicate with an appropriate SMTP system to receive these notifications.

Assess

Contrast Assess is an application security testing tool that combines Static (SAST), Dynamic (DAST), and Interactive Application Security Testing (IAST) approaches to provide highly accurate and continuous information on security vulnerabilities in your applications.



Contrast Assess uses an agent that instruments applications with sensors. The sensors look at data flow in real time and analyze the application from within to help figure out vulnerabilities in:

- Libraries, frameworks, and custom code
- Configuration information
- Runtime control and data flow
- HTTP requests and responses
- Back-end connections

Assess is appropriate for environments such as a test, QA, or staging servers. It is also applicable to developer workstations. When coupled with Contrast integrations, such as Visual Studio, developers can find and fix vulnerabilities without leaving their integrated development environment (IDE).

Features

Once you [install and configure an agent \(page 52\)](#) and [enable Assess \(page 1153\)](#), Contrast offers you these features:

- A list of [vulnerabilities \(page 1021\)](#) in the application, along with remediation guidance.
- [Application scores \(page 1269\)](#) to gauge the security of an application at a glance.
- [Route coverage \(page 619\)](#) that detects possible routes by associating vulnerabilities with the originating web request.
- [Flow maps \(page 631\)](#) that provide insight into the architecture of the running application.
- [Compliance and policy reporting \(page 1043\)](#).

Customization

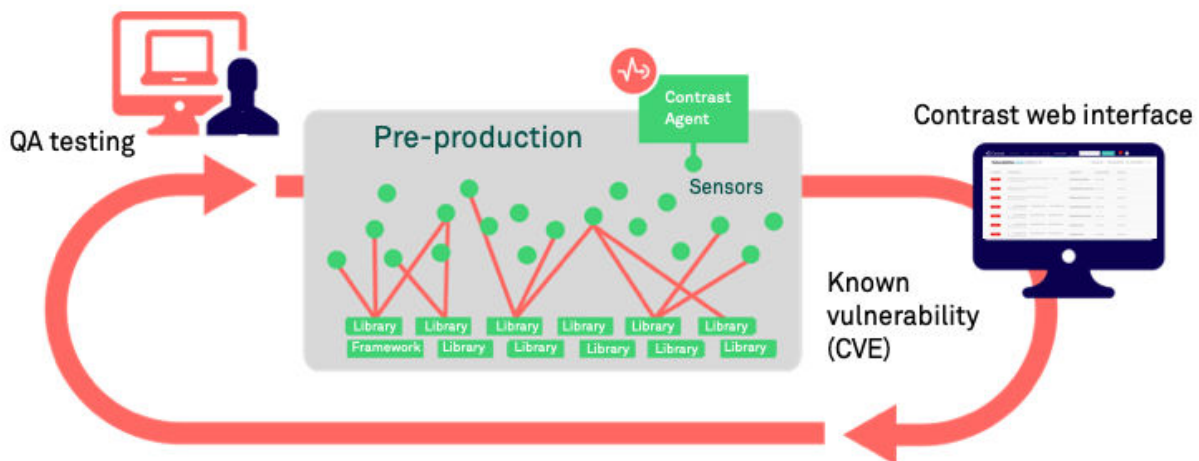
To customize Assess for your needs, you have the option of configuring these types of policies:

- [Assess rules \(page 1119\)](#) that you can enable or disable to fine tune the detection capabilities of Assess.
- [Security controls \(page 1120\)](#) are methods in your code that make sure data is safe to use.

SCA

Contrast SCA identifies open-source components through run-time analysis, file system scanning, and dependency analysis. Leveraging these techniques, SCA reports an exact inventory to Contrast.

By default, Contrast Assess includes powerful SCA capabilities. With an SCA license, you have access to advanced SCA capabilities.



Features

To simplify the process and merge open-source analysis with custom code analysis, SCA is integrated as part of the Contrast platform. Here's what you can do with SCA (some of these features are free and other require an SCA license):

- **Open-source license management:** Contrast SCA provides [license data \(page 933\)](#) tied to open-source components. This data helps you understand intellectual property compliance and mitigate operational risk.
This feature requires an SCA license.

- **Open-source policy:** With SCA, you can set policies to denylist open-source licenses. If a denylisted license type is deployed in your applications, it triggers an alert. To keep your library usage safe, [set compliance policies \(page 1147\)](#) for your organization. To restrict use of specific open-source libraries and licenses, as well as set version requirements, you can [set library policies \(page 1151\)](#). This feature requires an SCA license.
- **Identification of CVE vulnerabilities** Contrast SCA identifies the CVE vulnerabilities for each library that your applications are using. This data includes a description of each CVE vulnerability for a selected library as well as the number of applications using that library. This feature is available without an SCA license.
- **CLI and dependency tree:** The Contrast CLI performs software composition analysis (SCA) on your application to show you the dependencies between open source libraries, including where vulnerabilities were introduced. The data that the [Contrast CLI \(page 1014\)](#) collects is used to display a [dependency tree \(page 934\)](#) that brings awareness to underlying library dependencies. This feature is available without an SCA license.
- **GitHub action:** Use this integration to analyze a project's dependencies for vulnerabilities. The action will run Contrast SCA Action to detect vulnerable libraries. See [Contrast SCA Action](#) for more information.
- **Repository scans:** [Connect a repository \(page 935\)](#) to Contrast to scan for vulnerabilities.

Contrast data

Once a library is reported to Contrast, you can access:

- Library usage analysis to identify whether vulnerable components are actually used by the application
- Library version identification and guidance on the latest version
- Comprehensive vulnerabilities coverage
- Portfolio wide, real-time reporting of open-source components

Static SCA

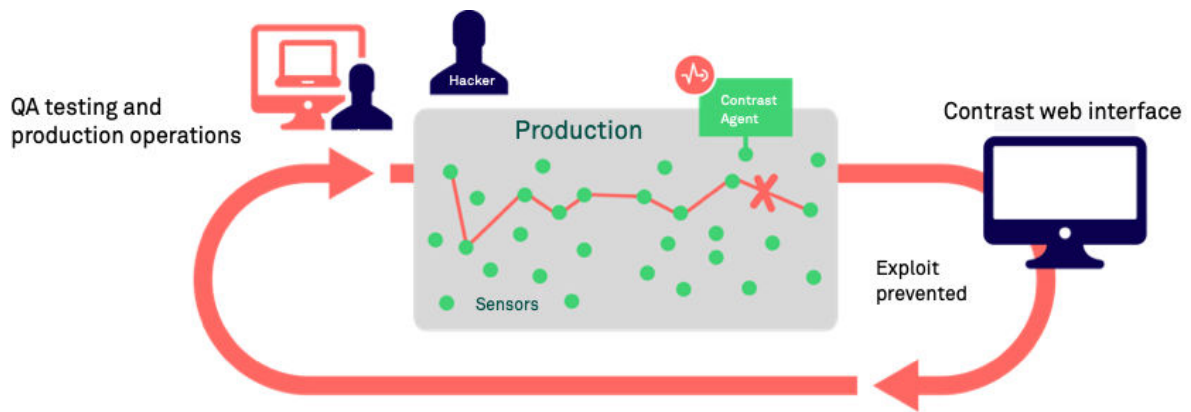
As of June 2024, Static SCA is an opt-in feature. Contrast will support current users by resolving reported bugs.

Protect

Protect is a defensive control for production environments that monitors attacks and actively defends applications based on specific vulnerabilities, for example, command injection.

It offers Runtime Application Self-Protection (RASP) that complies with NIST 800-53, PCI-DSS, PCI-SSS, and other industry standards. Protect operates directly inside runtimes such as [Java \(page 120\)](#), [.NET \(page 212\)](#), [.NET Core \(page 270\)](#), [Node.js \(page 329\)](#), [Ruby \(page 468\)](#), and [Python \(page 409\)](#), to leverage in-app intelligence without any manual tuning.

Contrast Protect blocks both automated and advanced threats attacking web applications and API, and provides valuable and timely application layer threat intelligence across the entire application portfolio.



How Protect works

Contrast Protect works inside application software to understand complete data flow rather than network traffic. Instead of only analyzing incoming data, Protect sees the same data and watches its impact on underlying actions, such as complete SQL queries, command arguments, and more.

This analysis improves detection accuracy, separating the noise of many attacks that might be false positives to focus on attacks that met their intended target. This insight can be shared with external systems, such as a SIEM, to focus on key attack events.

Protect limits its impact on application performance by operating with the same shared memory as the application to avoid additional overhead. Contextual defense improves performance by avoiding unnecessary actions. For example, NoSQL applications do not need checks against SQL injection if the SQL APIs are never invoked.

Customization

When Protect is enabled, you can customize these policies and rules:

- **Protect Rules:** (page 1130) Set applications to monitor for attacks.
- **CVE shields:** (page 1134) Specify CVE shields that block vulnerabilities.
- **Virtual Patches:** (page 1138) Define custom defenses against specific vulnerabilities.
- **Log Enhancers:** (page 1140) Provide additional instrumentation instructions.
- **IP Management:** (page 1149) Manage a denylist and allowlist (trusted hosts).

See also

[Agent performance with Protect \(page 573\)](#)

Contrast Protect licensing guide

This guide describes the Contrast licensing model for Protect. It explains key terms and provides real-world examples to illustrate how Contrast applies Protect licensing.

The goal of this guide is to provide clarity about the licensing model for technical and non-technical audiences.

Assess versus Protect licenses

- **Assess licenses:** These licenses apply to individual applications, regardless of the number of times they are running.
- **Protect licenses:** These licenses are based on the number of servers in the production environment. The licenses can adjust based on the changing number of servers in use. Contrast allows for burst flexibility; license use can go beyond the number of licenses purchased. This option allows for dynamic scaling of applications and ensures Contrast is able to provide continuous service with minimal interference.

Common scenarios

- **One application to one server**

If you are running one application on one server, you need one Protect license. This scenario is the most basic one.

- **One application across multiple servers**

If you have one application distributed across several servers, each server requires its own Protect license. This scenario is typical in high-availability or load-balanced configurations.

- **Multiple applications on one server**

If you are running several applications on a single server, you need just one Protect license for that server. This scenario is common in environments that use Java Virtual Machines (JVMs) or Microsoft Internet Information Services (IIS), where multiple applications coexist on a single server.

Example: StreamFlix, a Video Streaming Platform

This real-world example shows how you could use Protect licensing.

Background

StreamFlix operates a video streaming platform and employs a microservices architecture. They have several microservices, including User Authentication, Video Playback, and Analytics. StreamFlix uses ephemeral servers that can scale up or down based on demand.

Event

A highly anticipated series premiere is scheduled. As users start logging in to watch, there's a sudden surge in traffic.

Microservice behavior

Phase	User authentication service	Video playback service	Analytics service
Normal operation	5 servers	10 servers	3 servers
Premiere day	20 servers Due to the influx of users logging in, the number of servers scale up to handle demand.	50 servers More users are streaming.	5 servers More servers are required to process the higher volume of user data.
Post premiere	5 servers As traffic normalizes, the number of servers scales down.	15 servers Once the rush subsides, the numbers of servers scales down to a slightly higher number due to continued interest.	3 servers

Protect licenses

Phase	Protect licenses
Normal operation	18 licenses (5 + 10 + 3 servers)
Premiere day	License use surges to 75 (20 + 50 + 5 servers)
Post premiere	License use adjusts to 23 (5 + 15 + 3 servers)

Key insights

- **Adaptive Licensing:** Protect licensing dynamically adjusts to the changing number of servers during high-demand events, ensuring continuous service.
- **Distinct Focus:** Asses licensing focuses on unique applications. Protect licensing focuses on the servers running these applications in a production environment.

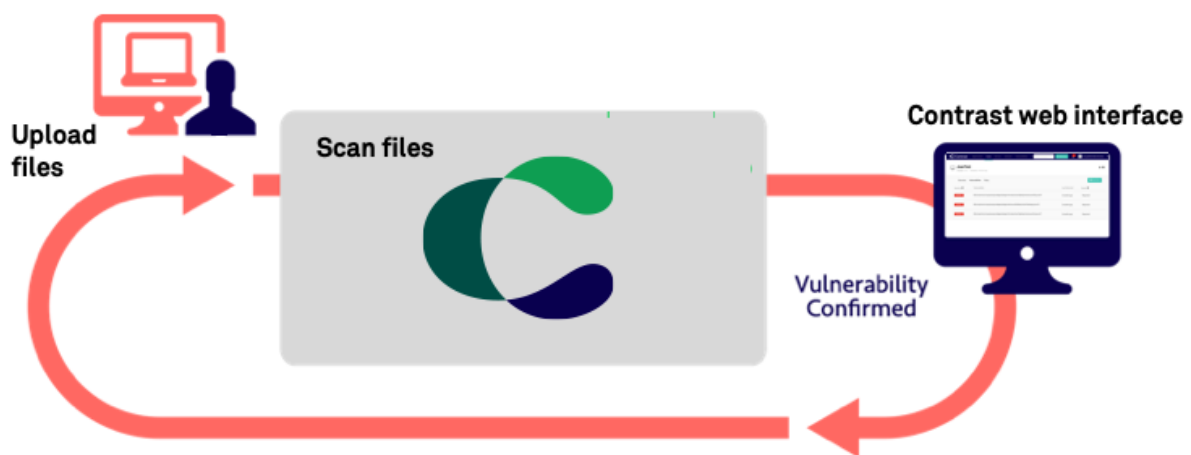
Scan

[Contrast Scan \(page 632\)](#) is a static application security testing (SAST) tool that makes it easy for you to find and remediate vulnerabilities. It is a valuable tool to use during the development phase of an application. Licensed, hosted customers have access to this feature.

To scan an application, you upload binary packages to a Contrast secure environment. After you upload the code, you start the scan. The scan observes the data flows in the source code and identifies vulnerabilities that could allow malicious attacks. Some examples of these malicious attacks include SQL injections, command injections, and server-side injections.

The scan results identify vulnerabilities in custom code. After fixing these issues, running the scan again verifies that the code changes removed one or more vulnerabilities.

No open-source code or libraries are included in the scan.



Features

- Ability to create scan groups that enable you to track results of multiple scans
- Scan settings that let you change the name of a scans
- Starting or stopping scans
- Views of identified vulnerability details
- Monitoring of scan progress and history
- Assignment of status to vulnerability records
- Integration of scanning into your CI/CD pipeline
- Information about risk and approaches for fixing each type of vulnerability

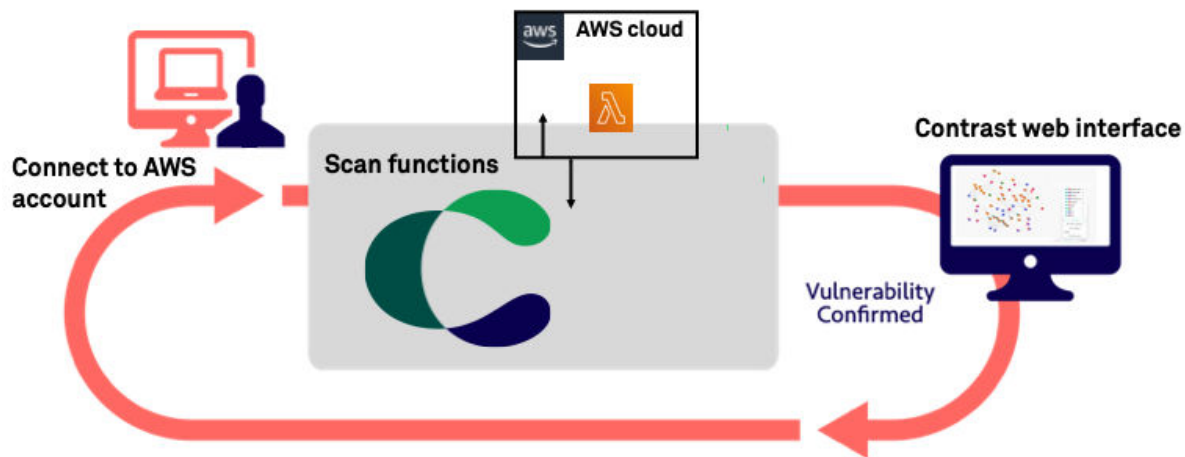
See also

[Scan supported languages \(page 657\)](#)

Serverless Application Security

Contrast Serverless Application Security is a next-generation application security testing solution for serverless-based applications.

Contrast Serverless Application Security uses cloud-native architecture to map all resources within your environment, while automatically validating and prioritizing the results, eliminating false-positive results and alert fatigue. It scans for vulnerabilities in your custom code (for example, injection attacks), dependencies (for example, CVEs), and configuration risks (over-permissive function policies).



Features

- **Easy connection to AWS**
With two clicks and approximately two minutes, you connect to your AWS account through the Contrast web application.
- **Discovery of your inventory**
Once you connect to your AWS account, Contrast creates an inventory of your functions, resources, policies, and services in your AWS environment.
- **Analysis of vulnerabilities**
Dynamic and static scans analyze your code, discovering weaknesses, data flows, attack surfaces, and exposure to vulnerabilities.
- **Continuous monitoring**
As your code changes, Contrast continues to monitor your Lambda functions, identifying vulnerabilities that require attention.
- **Simulation of attacks**
A dynamic scan generates and executes curated attacks on resources and data flows, without making changes to your code.
- **Visual representation of function and service relationships**
In addition to viewing relationships between functions and services in your account, you can view details about each element including applicable risks.
- **Reporting**
Scan results list CVEs, permission violations, vulnerabilities, and other exposures in your code.

Benefits

- **Fast and easy deployment**
You do not need a large staff of specialists or consultants or a lot of time to integrate with Contrast.
- **Non-intrusive integration**
You can add serverless security without significant changes to the development process. It's easy to find and fix exploitable vulnerabilities early in the development phase, ensuring that your applications are more secure when you deploy them to production.

How it works

Contrast connects to your AWS account with ReadOnly access. It uses this access to continuously monitor the environment and collect relevant information.

Contrast deploys one Lambda function (Cloud Agent) within the monitored environment, to perform activities such as code analysis and sending back to Contrast meta-data about the scanned resources (Lambda functions).

All the information is used by our contextual engine to build a tailored attack profile for every resource and change to this environment. The attack simulations will be executed, inside the customer's account, by the Cloud Agent.

All finding results will go through an internal validation mechanism to qualify them, providing zero false-positive and real prioritized results.

Security and privacy

- Contrast does not collect code or code-snippets from your monitored account. Contrast only sends back meta-data information such as:
 - Identified vulnerabilities
 - Function names and metadata (for example, policy handlers)
 - Used libraries
 - AWS API calls (for example, boto3 and asw-sdk)
 - Service configurations (for example, bucket notifications, API gateway paths, and methods)
- Contrast makes no changes to your code. However, during the scan time (for example, when a function is deployed or modified), Contrast temporarily instruments a layer into the scanned function and makes some configuration changes (for example, timeouts or handlers). Once the scan completes, Contrast restores the layer and the configurations to their original states. This process is completely transparent and occurs automatically. During a scan, you can continue to run your own tests (function calls).
- During dynamic scans, Contrast executes with scanned function using malicious data. This process has no effect on your code. It does, however, execute code that could potentially trigger any action that the function makes. This function is disabled by default. Use the Settings tab to enable it at any time.
- All data that Contrast receives from the monitored AWS account is encrypted in transit and at rest. Contrast uses Amazon EventBridge with a shared secret to send and receive all data. There are no web or REST APIs that Contrast uses to communicate with your AWS account.

Requested permissions

When you use Contrast to connect to your AWS account, you consent to these access permissions:

- ReadOnlyAccess from the Contrast AWS account to your monitored AWS account. This policy is used during beta activities only.
- Lambda Read/Write access from the Contrast AWS account to the Lambda functions deployed in your AWS account.
- The Lambda function that Contrast installs in your monitored AWS account requires these access policies:
 - Read CloudWatch Logs
 - Read Layer versions
 - Invoke function
 - Change function configuration
 - Write EventBus messages
 - Read KMS keys
 - Read/Write objects to specific S3 buckets (Contrast creates these buckets)

See also

- [Get started with Contrast Serverless \(page 968\)](#)

Contrast performance and resource consumption

Minimize the impact of Contrast on production servers by using the proper configuration:

- **Development environments:** Contrast Assess should be on and Protect can be off. This provides the strongest insight into an application's security posture. This detailed insight favors deep insight over performance to focus on helping developers locate security flaws.
- **Test environments:** Contrast Assess or Protect should be enabled based on what the team needs. Teams should strike a balance to achieve the overall goals of the team:
 - If little testing is done in development, teams should leverage Assess to find vulnerabilities as the application is used.
 - When evaluating performance, Contrast Assess should be turned off and only Protect should be enabled. This provides a corrective control that favors performance but still retrieves code-level information when corrective action is needed.
- **Production environments:** Only Contrast Protect should be on. This provides contextual defense while favoring performance.

See also

[Agent performance with Protect \(page 573\)](#)

[System requirements for .NET Framework agent \(page 213\)](#)

[.NET Core system requirements \(page 272\)](#)

[System requirements for the Node.js agent \(page 333\)](#)

[PHP agent system requirements \(page 392\)](#)

[System requirements for the Python agent \(page 410\)](#)

[System requirements for the Ruby agent \(page 470\)](#)

Community Edition (CE)

Community Edition offers near full access to Contrast products (Assess, SCA, and Protect), with developers receiving interactive application security testing (IAST), software composition analysis (SCA), and runtime application self-protection (RASP) solutions—all for free.

[Sign up for a Community Edition account](#) and install an agent to get started. Learn more on the [Community Edition blog](#).



NOTE

Community Edition lets you add **one** Java, .NET Core, or Node.js application to Contrast.

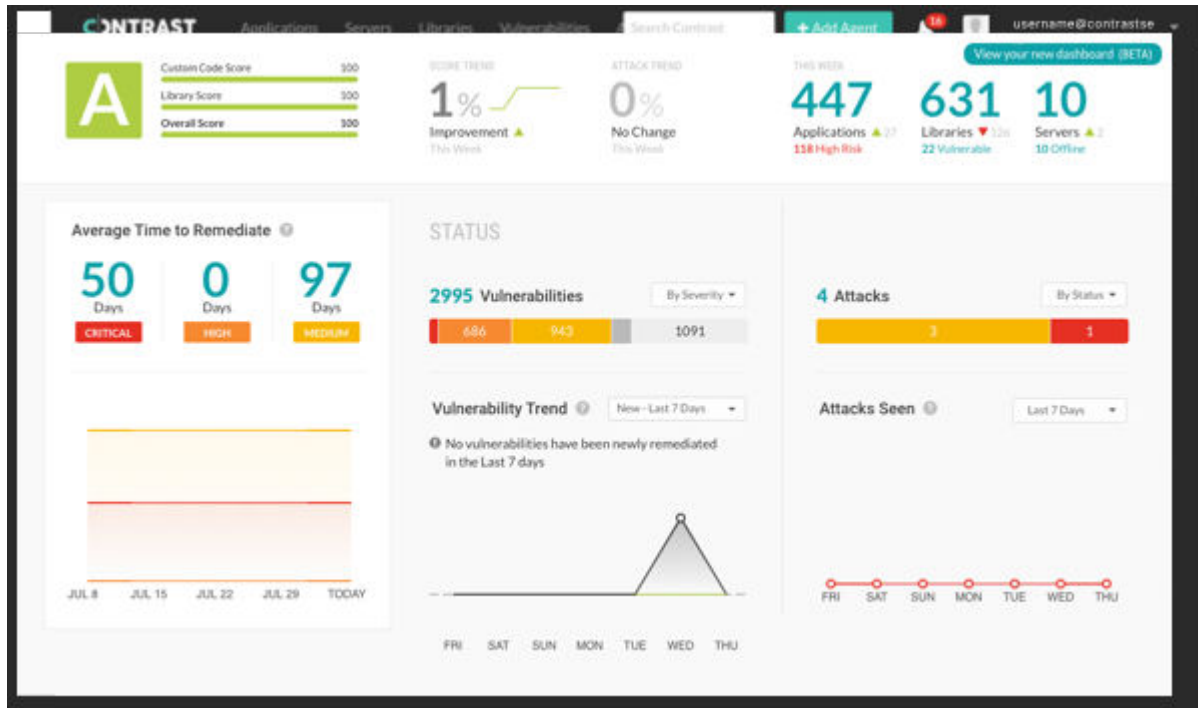
Community Edition features

Community Edition offers:

- Contrast Assess, which allows developers to focus only on fixing vulnerabilities derived from custom code that actually matter.
- Contrast SCA, which delivers unparalleled visibility into and management of security risks from vulnerabilities introduced through open-source and third-party libraries
Contrast SCA is an open-source security or software composition analysis (SCA) solution.
- Contrast Protect, which monitors and automatically blocks attacks on applications using instrumentation from within the application— even if the vulnerability still exists in self-written code or open-source libraries.

Community Edition portal

Here's an example of the CE portal that you interact with when using Contrast:



Next steps

- [Install the .NET Core agent \(page 273\)](#)
- [Install the Java agent \(page 123\)](#)
- [Install the Node.js agent \(page 334\)](#)

Contrast for developers

The Contrast platform

Your first step is understanding the technologies in the Contrast platform so you can choose an appropriate analysis strategy.

Code analysis during development

- **Contrast Scan**

Contrast Scan is a static application security testing (SAST) tool that lets you quickly scan code to identify vulnerabilities in early stages of development.

- **Why use Contrast Scan?**

Contrast Scan provides exceptional speed without sacrificing accuracy. It takes just a few minutes and clicks to start a scan. In addition, you can use a local scan engine to avoid uploading your content to the Contrast platform directly. Contrast Scan is a good choice for client-side code such as Angular, React, or Vue.js based applications.

- **Contrast Serverless**

For functions as a service (FaaS)-style serverless, Contrast Serverless protects you in a number of ways:

- It does some static and dynamic analysis to detect vulnerabilities.
 - It does some SCA analysis for open source libraries.
 - It also analyzes your functions to determine the least privilege configuration necessary for your serverless functions to operate but closes off avenues for attackers.

For serverless offerings like AWS Fargate, you can use Contrast Assess. For Azure Functions, you can use both Contrast Serverless and Contrast Assess. You can use the Contrast CLI to access a subset of the functionality, which also provides you with a pipeline integration option. However, the primary mode for Contrast Serverless is identifying and protecting all the functions in your cloud-provider account with only a few clicks and a few minutes worth of work.

- **Why use Contrast Serverless?**

- It provides you with a comprehensive list of your functions and enables you to make them secure from attack
 - No additional time or retooling of DevOps pipelines is required to benefit from serverless function scanning
 - No extra overhead is needed to look at invalid data
 - Contrast Serverless assists you in making your code more secure by guiding you to select appropriate policies.

Analysis of open source libraries

Contrast SCA

Contrast has always provided runtime Software Composition Analysis (SCA) capability with Assess, but now, you can also use Contrast SCA to detect vulnerabilities in your 3rd party dependencies (mostly open source) statically using a command line interface (CLI) and through our GitHub integrations, that allows bulk onboarding of projects to Contrast

Why use Contrast for SCA instead of other good (and often free or inexpensive) SCA tools?

Contrast SCA lets you focus only on what matters. Contrast runtime SCA provides a unique ability to not only tell if your application dependency manifests specify a vulnerable version of a vulnerable library but it can tell you which libraries are actually invoked, and to what degree, at runtime. This ability lets you lower the priority on the 70% that aren't invoked at runtime. Contrast SCA can also detect libraries

that are not listed in the manifests but injected at runtime by the environment -- a blindspot for pure static SCA solutions like most free ones.

Code analysis during runtime

Contrast Assess

Contrast Assess is the *interactive application security testing* (IAST) part of the Contrast platform. The core of an IAST tool is sensor modules, software libraries included in the application code. These sensor modules keep track of application behavior while the interactive tests are running. IAST analyzes code in runtime to find vulnerabilities, like *static application security testing* (SAST) tools do prior to compile and execution. It analyzes runtime behavior, like *dynamic application security testing* (DAST) tools. It also serves as the collector for our runtime *Software Composition Analysis* (SCA) capability. So, you can think of Contrast Assess as four tools in one.

Why use Contrast Assess?

Assess has a fraction of the false positives while finding up to twice the true positives as other SAST tools, without adding any additional scan wait times for both SAST and DAST tools. That's because with Contrast Assess, each interaction of the application by a user or your automated QA tests raises valuable telemetry about the security of the code in operation. This information makes IAST the simplest and least intrusive security process to add earlier in the development cycle, since no changes to your process are needed and will add no delays to your release schedule

Protection for production builds

Contrast Protect

Contrast Protect, is a *runtime application self-protection* (RASP) tool. Using the same technology as Assess, it blocks traffic that would have resulted in a successful attack.

Why should you integrate Protect into builds that you deploy to production?

Contrast Protect provides negative-day protection for zero-day attacks. The three-year-old version of Protect was able to block the infamous Log4Shell remote code execution attacks. For actively developed applications, this gives you time to upgrade, avoiding the late-night rush when a new zero-day emerges.

For maintenance-mode applications, it might serve as a long-term protection allowing you to stay focused on new applications. It also provides developers with actionable threat intelligence because it not only tells you attacks are occurring (information you may or may not be getting from your security team) but it also shows you the code paths used by the blocked attackers so you can easily eliminate them.

Next step

[Review analysis paths \(page 39\)](#)

Contrast analysis paths

Contrast provides multiple ways to you integrate secure code analysis into your development workflow.

This topic presents suggested paths for integrating Contrast analysis into your development workflow. Your environment might require different workflows.

Analysis paths during development

Static scanning with Contrast Scan	➔	Use any of these:	➔	Get results from:	➔	Integrate with your CI/CD system (page 50)
		<ul style="list-style-type: none"> Contrast CLI (page 42) Contrast GitHub action (page 45) Contrast web interface (page 46) 		<ul style="list-style-type: none"> SARIF files (page 48) Contrast CLI (page 47) Contrast web interface (page 48) 		
Static and real-time function scanning with Contrast Serverless	➔	Use any of these:	➔	Get results from:	➔	Integrate with your CI/CD system (page 50)
		<ul style="list-style-type: none"> Contrast CLI (page 42) Contrast web interface (page 46) (scan set up only) 		<ul style="list-style-type: none"> Contrast CLI (page 47) Contrast web interface (page 48) 		

Analysis paths for open source libraries

Static scanning with Contrast SCA	➔	Use any of these:	➔	Get results from:	➔	Integrate with your CI/CD system (page 50)
		<ul style="list-style-type: none"> Contrast CLI (page 41) GitHub app BitBucket or GitLab repo connections (page 44) 		<ul style="list-style-type: none"> Contrast CLI (page 47) Contrast web interface (page 48) Workflow integrations (page 1051) 		
Run-time scanning with Contrast SCA	➔	Use this:	➔	Get results from:	➔	Integrate with your CI/CD system (page 50)
		Instrumentation with Contrast agents (page 45)		<ul style="list-style-type: none"> Contrast web interface (page 48) Workflow integrations (page 1051) 		

Analysis path during runtime

Find application vulnerabilities with Contrast Assess	➔	Use any of these::	➔	Get results from:	➔	Integrate with your CI/CD system (page 50)
		<ul style="list-style-type: none"> Instrumentation with Contrast agents (page 46) Contrast CLI (page 43) 		<ul style="list-style-type: none"> Contrast CLI (page 47) Contrast web interface (page 48) IDE integrations (page 47) Workflow integrations (page 1051) 		

Protection path for production builds

Protect production builds with Contrast Protect	➔	Use this:	➔	Get results from:
		Instrumentation with Contrast agents (page 49) Contrast GitHub action (page 45)		<ul style="list-style-type: none"> Contrast web interface (page 48)

Code analysis

Once you decide on your approach for code analysis, you're ready to start testing your open source and source code.

To analyze...	With...	Go to:
Open source libraries	CLI	Use CLI for open source library analysis (page 41) (static analysis)
	GitHub app	Use GitHub app for open source library analysis (page 44) (static analysis)
	BitBucket connection	Connect repos for open source library analysis (page 44) (static analysis)
	GitLab connection	Connect repos for open source library analysis (page 44) (static analysis)
	Application instrumentation	Instrument applications for open source library analysis (page 45) (runtime analysis)
Source code	CLI	<ul style="list-style-type: none"> • Use CLI for static scanning (page 42) • Use CLI for serverless function scanning (page 42)
	GitHub action	Use GitHub action for static scanning (page 45)
	Application instrumentation	Instrument applications to find vulnerabilities (page 46)
	Contrast web interface	<ul style="list-style-type: none"> • Use Contrast web interface for static scanning (page 46) • Use Contrast web interface to set up serverless function scanning (page 46)

Next steps

- [Get results from the CLI \(page 47\)](#)
- [Get results from IDE integration \(page 47\)](#)
- [See results in SARIF files \(page 48\)](#)
- [See results in the Contrast web interface \(page 48\)](#)

Use CLI for open source library analysis

The Contrast CLI lets you analyze open source libraries for vulnerabilities and returns the results.

By default, the CLI doesn't store the results locally. To maintain persistent data, use the `CLI---track` option to send the results to the Contrast web interface.

Before you begin

- Learn about the [Contrast CLI \(page 994\)](#).
- [Install the CLI \(page 995\)](#).

Steps

1. Store your Contrast credentials locally with this command in a terminal window:

```
contrast auth
--api-key <ContrastAPIKey>
--authorization <ContrastAuthorizationHeader>
--host <YourHostDomain>
--organization-id <ContrastOrganizationID>
```

Get the Contrast API key, the authorization header, and organization ID by logging into the Contrast web interface and selecting **user menu > User settings**.

2. Find vulnerable libraries by using this command in a terminal window:

```
contrast audit [option]
```

- Use the `--track` option to send persistent results to the [Libraries \(page 924\)](#) Static tab in the Contrast web interface.

- Use the `--file` option to specify a directory or file to audit.
[CLI commands \(page 1002\)](#) describes all the valid options for the `audit` command.

Next steps

- [Get results with the CLI \(page 47\)](#)
- [See results in the Contrast web interface \(page 48\)](#)

Use CLI for static scanning

Instead of using the Contrast web interface, you can use the CLI to scan your code.

Before you begin

- Learn about the [Contrast CLI \(page 994\)](#).
- [Install the CLI \(page 995\)](#).

Steps

1. Store your Contrast credentials locally with this command in a terminal window:

```
contrast auth
--api-key <ContrastAPIKey>
--authorization <ContrastAuthorizationHeader>
--host <YourHostDomain>
--organization id <ContrastOrganizationID>
```

Get the Contrast API key, the authorization header, and organization ID by logging into the Contrast web interface and selecting **user menu > User settings**.

2. Upload and scan a package by using this command in a terminal window:

```
contrast scan --file <FileName>
```

[CLI commands \(page 1009\)](#) describes all the valid options for the `scan` command.

Next steps

- [Get results with the CLI \(page 47\)](#)
- [See results in the Contrast web interface \(page 48\)](#)

Use CLI for serverless function scanning

Instead of using the Contrast web interface, you can use the CLI for scanning your serverless functions.

Before you begin

- Learn about the [Contrast CLI \(page 994\)](#).
- [Install the CLI \(page 995\)](#).

Steps

1. Store your Contrast credentials locally with this command in a terminal window:

```
contrast auth
--api-key <ContrastAPIKey>
--authorization <ContrastAuthorizationHeader>
```

```
--host <YourHostDomain>  
--organization id <ContrastOrganizationID>
```

Get the Contrast API key, the authorization header, and organization ID by logging into the Contrast web interface under **user menu > User settings**.

- Find vulnerabilities by using this command in a terminal window:

```
contrast lambda --function-name <function> [options]
```

- Use `--json` to return the response in a JSON format.
- Use `--verbose` to return extended information to the terminal window.
- [CLI commands \(page 1011\)](#) describe all the valid options for the `lambda` command.

Next steps

- [Get results with the CLI. \(page 47\)](#)
- [See results in the Contrast web interface. \(page 48\)](#)

Use CLI to find vulnerabilities

The Assess CLI lets you use Contrast Assess to display vulnerabilities in real time.

The following Contrast agents support the Assess CLI:

- Java
- Node.js
- .NET
- Python
- Ruby
- Go

Before you begin

- Learn about the [Contrast CLI. \(page 994\)](#)
- [Install the CLI. \(page 995\)](#)
- Verify you can use the Assess CLI with your application by checking the supported technologies for your agent.

Steps

- Install or update an agent.
- In a terminal window, enter the following command:

```
contrast assess
```

This command creates the agent configuration file that both the Assess CLI and the agent share. The default locations for the file are:

- **MacOS and Linux:** `/etc/contrast/contrast_security.yaml`
- **Windows** `%ProgramData%\Contrast\contrast_security.yaml`

To specify a different file location, use the `config-path` option. [CLI commands \(page 1002\)](#) describes all the valid options for the `assess` command.

- Run your application in your IDE or a second terminal window.
- Exercise your application.
- View the results in the terminal window where you entered the Assess CLI command.

Next steps

[Get results with the CLI. \(page 47\)](#)

See also

[Use Assess CLI with Java agents \(page 997\)](#)

[Use Assess CLI with Node.js agents \(page 999\)](#)

[Use Assess CLI with .NET agents \(page 998\)](#)

[Use Assess CLI with Python agents \(page 1000\)](#)

[Use Assess CLI with Ruby agents \(page 1000\)](#)

[Use Assess CLI with Go agents \(page 1001\)](#)

Use GitHub app for open source library analysis

The Contrast GitHub app lets you connect your GitHub repo with Contrast. Once you establish this connection, Contrast scans the open source libraries in selected repos to identify vulnerabilities.

Before you begin

- To connect to the GitHub app, you need the subdomain and host for your Contrast account (for example: `app.contrastsecurity.com`)

Steps

1. Log in to the Contrast web interface and select **Add New** in the header.
2. Select the Repositories card tab and then, select **Connect GitHub**.
3. When prompted to do so, specify where you want to install the app in GitHub.
4. Follow the displayed steps until you complete the final authorization in the Contrast web interface. The Projects list start populating from your GitHub repositories.
5. Add more repositories at any time by selecting **Add repositories**.

Next steps

[View results in the Contrast web interface. \(page 48\)](#)

Connect repos for open source library analysis

For open source library analysis, you can connect to the GitHub, BitBucket, or GitLab platform. Once you connect to any of these platforms, Contrast displays scan results in the Projects tab.



NOTE

Connections to Bitbucket and GitLab are available by request only. Contact [Contrast Support](#) to enable these connections.

Steps

1. In the Contrast web interface, select the **Projects** tab.

2. Select **Add repository**.
3. To connect to GitHub, select the **GitHub card**. This option uses the [Contrast Security GitHub App \(page 936\)](#) to connect with Contrast.
4. To connect to BitBucket, select the **BitBucket card**. This option connects to your [Bitbucket repository \(page 937\)](#).
5. To connect to GitLab, select the **GitLab card**. This option connects to your [GitLab repository \(page 937\)](#).
You must have a GitLab Owner or Maintainer role to write the required variables into a GitLab repository.

Use GitHub action for static scanning

The GitHub Contrast Scan Analyze action compares the code scanning analysis of a pull request (PR) with the last code scan analysis of the destination branch.

Before you begin

- You need the following information from the Contrast web interface, under **user menu > User settings**:
 - Your API key
 - User authorization header
 - Organization ID

Steps

1. Access the GitHub action in the [Contrast repository](#).
2. [Set up the action](#).

Next steps

- [Get results in a SARIF file \(page 48\)](#).
- [See results in the Contrast web interface. \(page 48\)](#)

Instrument applications for open source library analysis

Instrumenting an application with a Contrast agent identifies open-source libraries included in an application. Contrast identifies any vulnerabilities found in your libraries and also confirms if the library is used at runtime.

Steps

1. [Install and configure a Contrast agent \(page 52\)](#) for the language that corresponds to the language your application uses.
You can download agents from a package manager or repository.
You can install agents directly or use [integrations \(page 1051\)](#) that work with Contrast.
2. Run the application to verify that Contrast is working. For example, click on your application's web interface or send some API or CLI commands.

Next steps

- [See results in the Contrast web interface \(page 48\)](#).
- [See results in an IDE. \(page 47\)](#)

Instrument applications to find vulnerabilities

To find application vulnerabilities, you use Contrast agents to instrument your application. You have the option of using a Contrast extension with your IDE so you can see results and resolve vulnerabilities in the IDE.

Basic steps

1. [Install the agent \(page 52\)](#) to the local directory where the application is located.
2. [Configure the agent \(page 102\)](#) using a YAML file or set environment variables that include the Contrast connection data.
The [Agent configuration editor \(page 109\)](#) provides an easy method to configure the agent.
3. Start the application and exercise routes.

Steps for using a Contrast IDE plugin

1. Install the agent to the local directory where the application is located.
2. Configure the agent using a YAML file or set environment variables that include the Contrast connection data.
The [Agent configuration editor \(page 109\)](#) provides an easy method to configure the agent.
3. Start the application.
4. Configure a [Contrast IDE plugin \(page 1051\)](#) with the required connection information.
In this case, you need your personal key or API information found in the Contrast web interface, under **user menu > User settings > Profile**.
5. Exercise routes in the application.

Next steps

[Review vulnerabilities in the IDE. \(page 47\)](#)

Use Contrast web interface to set up function scanning

Use the Contrast web interface to set up scanning for serverless functions.

Steps

1. Log in to the Contrast web interface and connect to your account: [AWS \(page 968\)](#) or [Azure \(page 974\)](#).
2. Set up scanning:
 - a. Select **Serverless** in the header.
 - b. Select the account that contains the functions you want to scan.
 - c. [Select the functions you want to scan \(page 975\)](#).

Next steps

[See results in the Contrast web interface. \(page 48\)](#)

Use Contrast web interface for static scanning

The Contrast web interface makes it easy to scan your code.

Before you begin

- [Check the supported languages \(page 657\)](#) for scanning.
- Learn about [preparing packages \(page 659\)](#) for scanning.

- Locate the artifacts you want to scan.
- [Create a scan project \(page 665\)](#).

Steps

1. Log in to the Contrast web interface
2. Select **Scans** in the header.
3. Select the scan project for the file you want to scan.
4. Start the scan:
 - a. Select **New scan**.
 - b. Upload the file.

Next steps

[See results in the Contrast web interface. \(page 48\)](#)

Analysis results

You have multiple options for viewing results after analyzing your open source libraries and code:

- [Get results when you use the CLI \(page 47\)](#)
- [Get results in your IDE \(page 47\)](#)
- [Get results in a SARIF file \(page 48\)](#)
- [See results in the Contrast web interface \(page 48\)](#)

Get results from the CLI

The CLI returns results in the terminal window after you run commands. The CLI doesn't store these results. Depending on the commands you use, you can send results to the Contrast web interface or download results in a SARIF file.

Steps

1. If you use the `audit` command, use the `--track` option to send results to the Static view on the Libraries page in the Contrast web interface.
2. If you use the `scan` command, use the `--save` option to download a SARIF file to the current working directory.

Get results from IDE integration

If you use a Contrast IDE plugin, you can view the vulnerability information directly in your IDE environment.

The IDE plugins that Contrast supports includes:

- Eclipse
- IntelliJ
- Visual Studio
- Visual Studio Code
- Visual Studio for Mac

Before you begin

Instrument your application by [installing and configuring a Contrast agent \(page 52\)](#).

Steps

1. Find a [Contrast IDE plugin \(page 1051\)](#).
2. Follow the plugin set up instructions.

Get results in SARIF files

You can choose to get results from static scanning in a SARIF file instead of in a terminal window (if using the CLI). You can also download a SARIF file from the Contrast web interface.

Steps

1. If you are using the CLI for static scanning, use this command option to store results in a SARIF file:

```
contrast scan --save
```

This command downloads the file to the current working directory with a default name of `results.sarif`. You can view the file with any text editor.

2. If you are using the Contrast web interface, download the results to a SARIF (or CSV) file:
 - Select **Scans** in the header.
 - In the Scan project list, select a project.
 - At the end of the row for a scan, select the Download icon (⬇️). Results are available for download for up to five days after the scan completes.
3. If you are using the GitHub action for static scanning and want to view results in the Security tab in the repository, include this GitHub action in your setup:

```
- name: Upload SARIF file
  uses: github/codeql-action/upload-sarif@v2
  with:
    sarif_file: results.sarif
```

The SARIF file name must be `results.sarif`.

Get results in the Contrast web interface

In most cases, no matter how you integrate your development workflow with Contrast, you can see results from code analysis in the Contrast web interface.

Before you begin

- Log in to the Contrast web interface.

Steps

1. To view findings for open source library analysis, select **Libraries** in the header. This view shows all libraries across all projects (static) and applications (runtime). You can also view libraries for a specific application in the Libraries tab of that application.
 - In the Libraries list, to view details about specific vulnerabilities, select a name or a section in the vulnerability bar.
2. To view findings for open source libraries after you use the CLI to analyze manifest files or from a repo connection, select **Projects** in the header.
 - In the Projects list, to view details about specific vulnerabilities, select a name or a section in the vulnerability bar.
3. To view application vulnerability information, select **Applications** in the header.
 - In the Applications list, to view details about specific vulnerabilities, select a section in the vulnerability bar.

4. To view static scan details, select **Scans** in the header.
 - a. In the Scans list, select a scan project.
 - b. To view details about vulnerabilities, select the **Vulnerabilities** tab .
5. To view serverless function scan details, select **Serverless** in the header.
 - a. To view details about vulnerabilities, select the **Results** tab.
 - b. To view details about vulnerabilities for a specific function, select the function in the **Results** list

Monitor or block attacks

Contrast Protect lets you configure Contrast agents to identify and manage malicious attacks for applications in production environments.

As a developer, you can [configure an agent \(page 49\)](#) to use Contrast Protect rules to monitor and block attacks. The agent reports these results in the [Contrast web interface \(page 49\)](#).

Instrument applications for Protect

Turning on Protect lets you use Contrast agents to identify, monitor, or block attacks for your applications in production environments.

Before you begin

- Check with a SuperAdmin to verify that Protect is turned on for your organization.
- Check with an Organization administrator to verify that you have permissions to view Protect data.

Steps

1. [Install and configure a Contrast agent \(page 52\)](#) for the language that corresponds to the language your application uses.

You can download agents from a package manager or repository. You can install agents directly or use [integrations \(page 1051\)](#) that work with Contrast.

In the agent YAML file:

 - a. Turn on Protect.
 - b. Configure the modes for the Protect rules (monitor, block, or off)
2. Run the application to verify that Contrast is working. For example, click on your application's web interface or send some API or CLI commands.

Next step

[View attack data in the Contrast web interface \(page 49\)](#).

View attack data in the Contrast web interface

The Contrast web interface displays details about attacks that occurred in your production environments.

Steps

1. In the Contrast web interface, select **Attacks** in the header.
2. Explore the different tabs to view data about attacks that affected your applications.

Integration options for continuous integration/continuous delivery

Contrast provides options for integrating Contrast with your continuous integration/continuous delivery (CI/CD) pipelines. If you are not responsible for CI/CD automation, discuss these options with your DevOps team.

Option	Description
Azure Pipelines extension (page 1069)	Use the Azure Pipelines extension to configure tasks and release gates that can fail based on vulnerability information that Contrast reports.
Bamboo (page 1073)	The Contrast Bamboo plugin lets you configure profiles for connecting to Contrast and verify builds against vulnerability thresholds.
Circle CI	The Contrast Circle CI orb lets you query the Contrast API to check if vulnerabilities were found in your application. If vulnerabilities are found above a set threshold, you can fail the build.
GitHub	Add a step to a GitHub pipeline which acts as a security gate, based on results that Contrast reports. You can configure a Job Outcome Policy or a threshold to specify which vulnerabilities trigger the pipeline to fail.
GitLab	You can create a stage within a GitLab pipeline which acts as a security gate, based on results that Contrast reports. You can configure GitLab variables that specify which vulnerabilities trigger the stage to fail.
Gradle (page 1084)	The Contrast Gradle plugin lets you integrate the <code>Contrast.jar</code> file with your build. It's capable of authenticating to Contrast, downloading the latest Java agent, and verifying your builds.
Jenkins (page 1089)	The Contrast plugin for Jenkins lets you add application security gates to this pipeline. These gates contain criteria that can fail the Jenkins job for a vulnerable application with a build result like <code>Failure</code> or <code>Unstable</code> .
Maven (page 1103)	The Contrast Maven plugin can integrate Contrast Assess and Scan into your project's Maven build.

Agents

Contrast agents are responsible for gathering security relevant data from an application, analyzing that data, and reporting findings to Contrast when necessary. In specific situations, a Contrast agent can also take actions within an application to prevent exploitation or enable a security defense.

A Contrast agent gathers security relevant information using a variety of security instrumentation techniques, including code scanning, library scanning, [instrumenting an application \(page 52\)](#), configuration file scanning, and other techniques. Any security instrumentation technique that gathers information is a sensor.

Sensors generate events that snapshot information directly from within an application. For example, a sensor might capture an incoming HTTP parameter, or the details of a SQL query being made to the database. Some sensors may also take action if necessary to help strengthen defenses or block malicious activity, typically by throwing a security exception that causes a vulnerability to be bypassed.

Events generated by sensors are all reported to the tracking and analysis part of the agent. Over time, the analysis engine [receives events \(page 1032\)](#) from all over the code of the application and builds them into traces. The analysis engine watches these traces for patterns of behavior that represent a violation of the Contrast rules.

For example, the analysis engine might see a data flow like this:

- An incoming HTTP parameter event
- Then another event shows that parameter being appended to a SQL query
- Finally, another event shows that query being sent to a database

If the analysis engine sees that data flow without the proper defenses (escaping or parameterization), it recognizes that trace to match the Contrast rule for SQL injection reports it to Contrast. The vast majority of the analysis is done locally in the agent, which enables Contrast's scalability and performance.

Use the agents that matches the language of the application you want to instrument:

- [Java \(page 120\)](#) instruments Java web applications and web APIs running on your container.
- [.NET Framework \(page 212\)](#) instruments .NET web applications and APIs running on IIS.
- [.NET Core \(page 270\)](#) instruments applications and APIs running in the .NET Core runtime.
- [Node.js \(page 329\)](#) instruments Node.js web applications and APIs.
- [PHP \(page 391\)](#) analyzes PHP web applications at runtime for library usage and vulnerability detection.
- [Python \(page 409\)](#) instruments Django, Flask and Pyramid web applications.
- [Ruby \(page 468\)](#) instruments Ruby on Rails web applications.
- [Go \(page 530\)](#) instruments Go web applications for library support and vulnerability reporting.



NOTE

Contrast agents are supported for one year after release. Older agents may continue to function and remain compatible, but they are no longer fully supported.

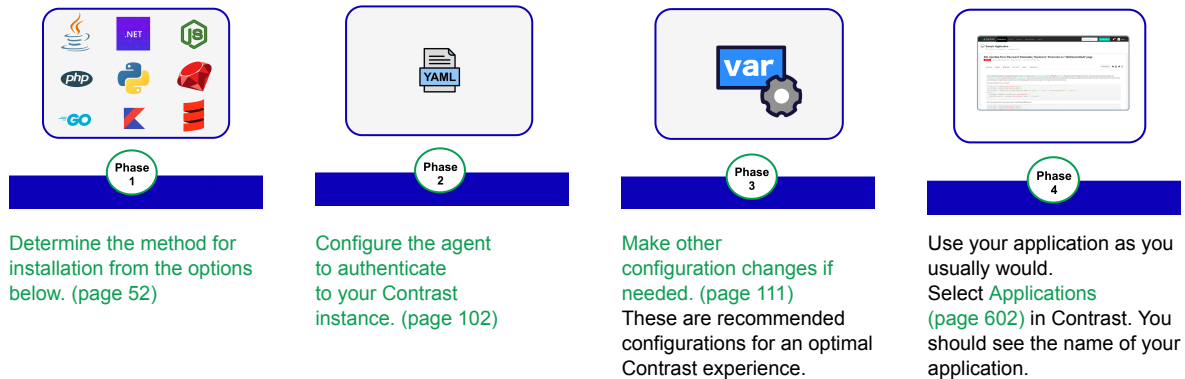
Contrast applies bug fixes and develops new features on the latest version of the agent. Code changes are not backported to previous versions. While a workaround may be provided for a bug, to resolve issues, you should update to the release in which the issue was addressed.

Install an agent

Contrast uses agents to install sensors that monitor your code for vulnerabilities. Agents analyze for vulnerabilities in development environments and look for attacks in runtime production environments.

As your application runs, the agent analyzes information (such as HTTP requests, data flow, backend connections, and library dependencies) and sends vulnerabilities and attacks to Contrast where you can view, prioritize, and take immediate action.

Instrumenting an app with Contrast can be divided into a few phases, so these guides should get Contrast up and running on your application in just a few minutes so you can see how it works.



Installation varies depending on the agent, which Contrast product(s) you are using, and where you want to install Contrast. For example, this could be:

- On an application server or web server
- In a build pipeline or container
- In a Develop, QA, or Production environment

Once you see how it works there are many ways to modify this to suit your needs. You can explore [Contrast Documentation \(page 51\)](#) for further information about how to adapt Contrast to your situation.



TIP

For future installations, you may want to consider your organization's build tools and deployment pipeline, your security goals and the environments where you want to use Contrast. You can read about [other methods to install Contrast \(page 1051\)](#) that may better adapt to your situation.

Java

View the [installation and configuration workflows \(page 55\)](#).

Install for executable JAR	Install to an app server	Install with build automation tool integrations	Install in a container	Install with cloud orchestration services	Install with infrastructure as code tools
<p>Install the agent in one application with a JAR file.</p> <p>Install with Maven Central (page 124), Debian (page 125), or RPM (page 126) repositories.</p>	<p>Install the agent to an app server to provide security analysis for applications running in a test/QA or production environment.</p> <p>For JBoss/Wildfly (page 143).</p> <p>For Jetty (page 144).</p> <p>For Tomcat (page 145).</p> <p>For Weblogic (page 146).</p> <p>For Websphere (page 147).</p> <p>For Axis2.</p> <p>For Glassfish.</p>	<p>Install the agent with Contrast plugins to automate the installation.</p> <p>For Maven (page 1103).</p> <p>For Gradle (page 1084).</p> <p>For Bamboo (page 1073).</p> <p>For Azure pipelines (page 1069).</p> <p>For VMware Tanzu (page 132).</p>	<p>Install the agent in a container image or via a Kubernetes operator.</p> <p>Add the agent to the Docker base or application image (page 127).</p> <p>For OpenShift.</p> <p>Add the agent to Kubernetes pods via Contrast k8s operator (page 552).</p>	<p>Install the agent for Google App Engine.</p> <p>Install agent with AWS Elastic Beanstalk (page 137).</p>	<p>Install the agent with infrastructure as code tools (page 130).</p>

You can also use the Contrast Java agent with Contrast Assess or Contrast SCA to analyze [Scala-based \(page 142\)](#) applications or to analyze [Kotlin-based \(page 142\)](#) applications.

.NET Framework

View the [installation and configuration workflows \(page 62\)](#).

Install with an installer	Install with Azure	Install in a container	Install with infrastructure as code tools
Install with an agent installer (page 214) for self-hosted applications or applications in IIS.	Install the agent with Azure App Service (page 217) .	Install the agent in a container image (page 220).	Install the agent with infrastructure as code tools (page 95) .

.NET Core

View the [installation and configuration workflows \(page 67\)](#).

Windows

Basic installation	Install with an installer	Install with Azure	Install in a container	Install with infrastructure as code tools
Install the .NET Core agent with the basic install (page 273) .	Install with an agent installer (page 281) for self-hosted applications or applications in IIS.	Install the agent with the Azure App Service (page 278) . Install the agent with Terraform (page 95) .	Install the agent in a container image or via a Kubernetes operator. Add the agent to the Docker base or application image (page 284) . Add the agent to Kubernetes pods via Contrast k8s operator (page 552) .	Install the agent with infrastructure as code tools (page 95) .

Linux

Basic installation	Install in a container
Install the .NET Core agent with the basic install (page 273) .	Install the agent in a container image (page 284) .

Node.js

View the [installation and configuration workflows \(page 74\)](#).

Basic installation	Install in a container	Install with Cloud deployment integrations	Install with infrastructure as code tools
Install the Node agent with the basic install (page 335) .	Install the agent in a container image (page 335) . Add the agent to Kubernetes pods via Contrast k8s operator (page 552) .	Install with IBM Cloud (page 344) . Install with VMware Tanzu .	Install the agent with Ansible playbook for Contrast (page 94) .

PHP

View the [installation and configuration workflows \(page 80\)](#).

Install by repository	Install in a container	Install to an app server
Install the PHP agent with the Debian (page 392) or RPM (page 394) repository.	Add the agent to Kubernetes pods via Contrast k8s operator (page 552) .	Install PHP agent on a Lando application server (page 396) .

Python

View the [installation and configuration workflows \(page 84\)](#).

Install with Contrast Runner	Install in a container	Install by middleware
Install and instrument the Python agent with the Contrast Runner (page 467) .	Add the agent to Kubernetes pods via Contrast k8s operator (page 552) .	Install the Python agent with AIOHTTP, or Bottle, or Django, or Falco, or Fast API, or Flask, or Pyramid, or Quart, or WSGI middleware (page 411) .

Ruby



IMPORTANT

The Ruby Agent is not currently being sold to new customers. Please contact our Sales team if you have any questions about our offerings and support for the Ruby language.

View the [installation and configuration workflow \(page 89\)](#).

Install by middleware
Install the Ruby agent with Rails or Sinatra middleware (page 471) .

Go

View the [installation and configuration workflow \(page 90\)](#).

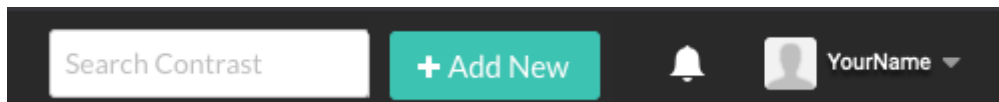
Install with an installer
Install the Go agent with the Contrast installer (page 531) .

Download an agent configuration file

You can download an agent's YAML configuration file that is pre-populated with required settings.

Steps

1. In the Contrast web interface, select **Add new**.



2. Select the Application tile.
3. Select the application language.
4. Download the pre-populated YAML file.

Java installation and configuration workflows

Follow these workflows to make sure you have covered the steps for installing and configuring the Java agent.

- [Java with JAR files \(page 55\)](#)
- [Java to an app server \(page 57\)](#)
- [Java with build automation tool integrations \(page 58\)](#)
- [Java in a container \(page 59\)](#)
- [Java in pipelines \(page 61\)](#)
- [Java with infrastructure as code tools \(page 130\)](#)

Java installation and configuration with JAR files

Use this workflow with the Add New Java agent wizard to ensure you have all the steps covered for installing and configuring the Java agent with an executable JAR file. Using the wizard is the simplest way to deploy a new agent.

Before you begin

Make sure you have everything you need before you start.

- The agent will need to be able to reach your Contrast instance. It can be a local/on-premise instance or a hosted instance. A proxy can be configured if the environment has limited network access.
- Your web application is packaged in a JAR file
- It must use [supported versions, frameworks, and tools \(page 120\)](#)
- Understand the [order of precedence \(page 106\)](#)
- You will also need access to a command line interface (with a chosen directory for downloading the agent) and your organization's instance of Contrast

Start the Java agent wizard

1. Select **Add New** at the top of the Contrast web interface.
2. Select the **Application** card.
3. Under Select a language, select **Java**.
4. Select the operating system you are using.

Install the agent

1. Under Select application deployment method, select **Install manually**.
2. Under Select agent installation method, select any of these options:
 - **Direct download:** Use the curl command in the Java agent wizard to download the `contrast.jar` file and install the agent.
 - **Debian:** Follow the steps in the Java agent wizard to get the Debian repository packages and install the agent.
 - **RPM:** Follow the steps in the Java agent wizard to get the RPM repository packages and install the agent.

Configure the agent

Under Configure the agent, select the best options to get the configuration details that the agent needs to communicate with Contrast:



NOTE

You have the option of using the [Contrast Agent Configuration Editor \(page 109\)](#) to configure agent settings. Select **Use configuration editor** next to Use connection token or Use API configuration.

1. Select the agent key that you want to use.

If only one key exists, it's preselected.

[Configure agent keys \(page 1162\)](#) describes how to create multiple agent keys.

2. Optionally, select **Register the application**.
When you register an application, you can assign it to groups for access control. You can also assign tags and metadata to the application.
3. For agent versions 6.10.1 and later, select **Use Connection Token**.
This configuration method uses a single variable to specify authentication values. The Connection Token is the recommended configuration option.
Copy the displayed code and set the variable in a location where the agent has access to it.

4. For agent versions earlier than 6.10.1, select **Use API configuration**.
This configuration method uses individual variables to specify authentication values.
Copy the displayed code and set these variables in a location where the agent has access to them.
5. If you prefer to configure the agent using a YAML file, select **Download configuration**.
Copy the displayed code to generate a YAML file that contains your authentication settings.
6. Under **Select application server**, select **Executable Jar**.
7. Use the displayed command to load and run the agent based on where it's installed.
Replace `<ApplicationJarPath>` with the path to your application. For example: `./MyApplication.jar`.

Verify the agent deployment

1. Interact with your application as you normally would. For example, interact with your application's web interface or send API commands
2. Go to the Contrast web interface and select **Applications** in the header.
3. Verify that you see the name of your application in the list.
4. Select servers in the header.
5. Verify that you see the name of your local server in the list.

Java installation and configuration to an application server

Use this workflow with the Add New Java agent wizard to ensure you have all the steps covered for installing and configuring the Java agent on an application server. Using the wizard is the simplest way to deploy a new agent.

Before you begin

Make sure you have everything you need before you start

- The agent will need to be able to reach your Contrast instance. It can be a local/on-premise instance or a hosted instance. A proxy can be configured if the environment has limited network access.
- Your web application is packaged in a JAR file
- It must use [supported versions, frameworks, and tools \(page 120\)](#)
- Understand the [order of precedence \(page 106\)](#)
- You will also need access to a command line interface (with a chosen directory for downloading the agent) and your organization's instance of Contrast

Start the Java agent wizard

1. Select **Add New** at the top of the Contrast web interface.
2. Select the **Application** card.
3. Under **Select a language**, select **Java**.
4. Select the operating system you are using.

Install the agent

1. Under **Select application deployment method**, select **Install manually**.
2. Under **Select agent installation method**, select **Direct download**.
3. Copy the displayed command to download the agent from Maven Central.

Configure the agent

Under **Configure the agent**, select the best options to get the configuration details that the agent needs to communicate with Contrast:

**NOTE**

You have the option of using the [Contrast Agent Configuration Editor \(page 109\)](#) to configure agent settings. Select **Use configuration editor** next to Use connection token or Use API configuration.

1. Select the agent key that you want to use.

If only one key exists, it's preselected.

[Configure agent keys \(page 1162\)](#) describes how to create multiple agent keys.

2. Optionally, select **Register the application**.
When you register an application, you can assign it to groups for access control. You can also assign tags and metadata to the application.
3. For agent versions 6.10.1 and later, select **Use Connection Token**.
This configuration method uses a single variable to specify authentication values. The Connection Token is the recommended configuration option.
Copy the displayed code and set the variable in a location where the agent has access to it.
4. For agent versions earlier than 6.10.1, select **Use API configuration**.
This configuration method uses individual variables to specify authentication values.
Copy the displayed code and set these variables in a location where the agent has access to them.
5. If you prefer to configure the agent using a YAML file, select **Download configuration**.
Copy the displayed code to generate a YAML file that contains your authentication settings.
6. Under Select application server, select one of these application server options:
 - [JBoss/Wildfly \(page 143\)](#)
 - [Jetty \(page 144\)](#)
 - [Tomcat \(page 145\)](#)
 - [Weblogic \(page 146\)](#)
 - [Websphere \(page 147\)](#)
 - [Axis2](#)
 - [Glassfish](#)
7. Under Configure application server, copy the displayed commands to complete the configuration.

Verify the agent deployment

1. Interact with your application as you normally would. For example, interact with your application's web interface or send API commands
2. Go to the Contrast web interface and select **Applications** in the header.
3. Verify that you see the name of your application in the list.
4. Select servers in the header.
5. Verify that you see the name of your local server in the list.

Java installation and configuration with build automation tool integrations

Use this workflow to ensure you have all the steps covered for installing and configuring the Java agent with build automation tools.

Before you begin

Make sure you have everything you need before you start.

- The agent will need to be able to reach your Contrast instance. It can be a local/on-premise instance or a hosted instance. A proxy can be configured if the environment has limited network access.

- Your web application is packaged in a JAR file
- It must use [supported versions, frameworks, and tools \(page 120\)](#)
- Understand the [order of precedence \(page 106\)](#)
- You will also need access to a command line interface (with a chosen directory for downloading the agent) and your organization's instance of Contrast

Steps

1. **Set up variables.** Configure agent authentication configuration variables.
 - Set the minimum configuration values defined [here \(page 102\)](#)
 - Configure additional values (application metadata, session metadata)
2. **Continue with the plugin type.** Install and configure based on your plugin type.
 - [Maven \(page 1103\)](#). Remember to set the [usage goals](#).
 - [Gradle \(page 1084\)](#)
 - [Bamboo \(page 1073\)](#). Remember to configure the [vulnerability thresholds \(page 1074\)](#).
 - [Azure pipelines \(page 1069\)](#). Remember to [add a release gate to a pipeline \(page 1071\)](#).
 - [VMware Tanzu \(page 132\)](#)
3. **Verify.** To verify that Contrast is working, use your application as you usually would. For example, click on your application's web interface, or send some API commands.
Then in the Contrast web interface, select **Applications** in the header. You should see the name of your application.
You can also select **Server** in the header and you should see the hostname of your (local) server.

Java installation and configuration in a container

Use this workflow with the Add New Java agent wizard to ensure you have all the steps covered for installing and configuring the Java agent with in a container. Using the wizard is the simplest way to deploy a new agent.

Before you begin

Make sure you have everything you need before you start.

- The agent will need to be able to reach your Contrast instance. It can be a local/on-premise instance or a hosted instance. A proxy can be configured if the environment has limited network access.
- Your web application is packaged in a JAR file
- It must use [supported versions, frameworks, and tools \(page 120\)](#) or [supported technologies \(page 553\)](#) for the agent operator
- Understand the [order of precedence \(page 106\)](#)
- You will also need access to a command line interface (with a chosen directory for downloading the agent) and your organization's instance of Contrast

Start the Java agent wizard

1. Select **Add New** at the top of the Contrast web interface.
2. Select the **Application** card.
3. Under Select a language, select **Java**.
4. Select the operating system you are using.

Select method to deploy with containers

1. Under Select application deployment method, select **Install with a container**.
2. Under Select method to deploy with containers, select **Dockerfile** or **Kubernetes** (Linux or OS X only).
If you want to use Openshift, use [these instructions](#).

Dockerfile: Build image with the agent

1. Under Build image with agent, copy the displayed command and add it to your Dockerfile.
2. Copy the displayed command to build the image.

Dockerfile: Configure the agent

Under Configure the agent, select the best options to get the configuration details that the agent needs to communicate with Contrast:



NOTE

You have the option of using the [Contrast Agent Configuration Editor \(page 109\)](#) to configure agent settings. Select **Use configuration editor** next to Use connection token or Use API configuration.

1. Select the agent key that you want to use.

If only one key exists, it's preselected.

[Configure agent keys \(page 1162\)](#) describes how to create multiple agent keys.

2. Optionally, select **Register the application**.
When you register an application, you can assign it to groups for access control. You can also assign tags and metadata to the application.
3. For agent versions 6.10.1 and later, select **Use Connection Token**.
This configuration method uses a single variable to specify authentication values. The Connection Token is the recommended configuration option.
Copy the displayed code and set the variable in a location where the agent has access to it.
4. For agent versions earlier than 6.10.1, select **Use API configuration**.
This configuration method uses individual variables to specify authentication values.
Copy the displayed code and set these variables in a location where the agent has access to them.
5. If you prefer to configure the agent using a YAML file, select **Download configuration**.
Copy the displayed code to generate a YAML file that contains your authentication settings.

Kubernetes: (Linux and OS X only)

1. Under Select method to configure agent operator, select **Helm** (recommended) or **Manifest**.
2. **For Helm:**
 1. Under Download and set up files, copy the displayed commands to create a values files.
 2. Under Label the deployments, use the displayed commands to label your deployments.
 3. Under Install and deploy Helm chart, copy the displayed commands to complete the configuration.
3. **For Manifest:**
 1. Under Install operator, copy the displayed commands to install the Contrast agent operator.
 2. Under Configure operator, copy the displayed commands to configure the Contrast agent operator.
 3. Under Inject workloads, copy the displayed commands to configure an AgentInjector entity.
 4. Optionally, under Configure cluster agent and cluster, copy the displayed commands to configure these entities.

Verify the agent deployment

1. Interact with your application as you normally would. For example, interact with your application's web interface or send API commands
2. Go to the Contrast web interface and select **Applications** in the header.
3. Verify that you see the name of your application in the list.
4. Select servers in the header.
5. Verify that you see the name of your local server in the list.

Java installation and configuration with build pipelines

Use this workflow with the Add New Java agent wizard to ensure you have all the steps covered for installing and configuring the Java agent with build pipelines. Using the wizard is the simplest way to deploy a new agent.

Before you begin

Make sure you have everything you need before you start.

- The agent will need to be able to reach your Contrast instance. It can be a local/on-premise instance or a hosted instance. A proxy can be configured if the environment has limited network access.
- Your web application is packaged in a JAR file
- It must use [supported versions, frameworks, and tools \(page 120\)](#)
- Understand the [order of precedence \(page 106\)](#)
- You will also need access to a command line interface (with a chosen directory for downloading the agent) and your organization's instance of Contrast

Start the Java agent wizard

1. Select **Add New** at the top of the Contrast web interface.
2. Select the **Application** card.
3. Under Select a language, select **Java**.
4. Select the operating system you are using.

Install the agent

1. Under Select application deployment method, select **Install with a pipeline**.
2. Under Select method to deploy pipelines, select **Github** or **Bitbucket**.
3. Under Install Agent, copy the displayed commands to install the agent.

Configure the agent

Under Configure the agent, select the best options to get the configuration details that the agent needs to communicate with Contrast:



NOTE

You have the option of using the [Contrast Agent Configuration Editor \(page 109\)](#) to configure agent settings. Select **Use configuration editor** next to Use connection token or Use API configuration.

1. Select the agent key that you want to use.

If only one key exists, it's preselected.

[Configure agent keys \(page 1162\)](#) describes how to create multiple agent keys.

2. Optionally, select **Register the application**.

When you register an application, you can assign it to groups for access control. You can also assign tags and metadata to the application.

3. Select **Use Connection Token**.

This configuration method uses a single variable to specify authentication values. The Connection Token is the recommended configuration option.

Copy the displayed code and set the variable in a location where the agent has access to it.

Verify the agent deployment

1. Interact with your application as you normally would. For example, interact with your application's web interface or send API commands
2. Go to the Contrast web interface and select **Applications** in the header.
3. Verify that you see the name of your application in the list.
4. Select servers in the header.
5. Verify that you see the name of your local server in the list.

Chef cookbook for Contrast agents

The Contrast Chef cookbook automatically installs a Contrast agent in a specific directory under the ownership and permissions of a specified Contrast user.

The [Chef documentation](#) describes how to set up a Chef Server.

Requirements

- A Chef Server
- The [Contrast Chef cookbook](#) from the Chef Supermarket
- **Optional:** Knife configured on your workstation
The `knife` command let you communicate with the Chef Server from your workstation

Integration example

This example shows the steps you might take to add a Contrast recipe to your run-list.

1. Open the Chef management console.
2. Select **Nodes**.
3. Select a node.
4. Select **Edit Run List**.
5. In the Edit Node Run List dialog box, drag the role or recipe from the Available Roles or Available Recipes lists to the current run-list.
6. Select **Save Run List**.

.NET Framework installation and configuration workflows

Follow these workflows to make sure you have covered the steps for installing and configuring the .NET Framework agent.

- [.NET Framework with an installer \(page 63\)](#)
- [.NET Framework with Azure App Service \(page 64\)](#)
- [.NET Framework in a container \(page 65\)](#)
- [.NET Framework in a pipeline \(page 66\)](#)
- [.NET Framework with an infrastructure as code tool \(Ansible playbook\) \(page 94\)](#)

.NET Framework installation and configuration with an installer

Use this workflow with the Add New .NET Framework agent wizard to ensure you have all the steps covered for installing and configuring the .NET Framework agent with an installer. Using the wizard is the simplest way to deploy a new agent.

Before you begin

Make sure you have everything you need before you start.

- The agent will need to be able to reach your Contrast instance. It can be a local/on-premise instance or a hosted instance. A proxy can be configured if the environment has limited network access.
- It must use [supported technologies \(page 212\)](#) and [system requirements \(page 213\)](#)
- Understand the [order of precedence \(page 106\)](#)
- You will also need access to a command line interface (with a chosen directory for downloading the agent) and your organization's instance of Contrast

Start the .NET Framework agent wizard

1. Select **Add New** at the top of the Contrast web interface.
2. Select the **Application** card.
3. Under Select a language, select **.NET Framework**.
4. Select the operating system you are using.

Download the installer

Under Download the installer, copy the displayed shell commands and run them.

Configure the agent

Under Configure the agent, select the best options to get the configuration details that the agent needs to communicate with Contrast:



NOTE

You have the option of using the [Contrast Agent Configuration Editor \(page 109\)](#) to configure agent settings. Select **Use configuration editor** next to Use connection token or Use API configuration.

1. Select the agent key that you want to use.

If only one key exists, it's preselected.

[Configure agent keys \(page 1162\)](#) describes how to create multiple agent keys.

2. Optionally, select **Register the application**.
When you register an application, you can assign it to groups for access control. You can also assign tags and metadata to the application.
3. For agent versions 51.0.40 and later, select **Use Connection Token**.
This configuration method uses a single variable to specify authentication values. The Connection Token is the recommended configuration option.
Copy the displayed code and set the variable in a location where the agent has access to it.
4. For agent versions earlier than 51.0.40, select **Use API configuration**.
This configuration method uses individual variables to specify authentication values.
Copy the displayed code and set these variables in a location where the agent has access to them.

5. If you prefer to configure the agent using a YAML file, select **Download configuration**. Copy the displayed code to generate a YAML file that contains your authentication settings.

Run the installer

1. Extract the downloaded ZIP archive on the web server.
2. Run `ContrastSetup.exe` from the content directory located in the extracted directory.

Verify the agent deployment

1. Interact with your application as you normally would. For example, interact with your application's web interface or send API commands
2. Go to the Contrast web interface and select **Applications** in the header.
3. Verify that you see the name of your application in the list.
4. Select servers in the header.
5. Verify that you see the name of your local server in the list.

.NET Framework installation and configuration with Azure App Service

Use this workflow with the Add New .NET Framework agent wizard to ensure you have all the steps covered for installing and configuring the .NET Framework agent with a the Azure App Service. Using the wizard is the simplest way to deploy a new agent.

Before you begin

Make sure you have everything you need before you start.

- The agent will need to be able to reach your Contrast instance. It can be a local/on-premise instance or a hosted instance. A proxy can be configured if the environment has limited network access.
- It must use [supported versions, frameworks, and tools \(page 120\)](#)
- Understand the [order of precedence \(page 106\)](#)
- You will also need access to a command line interface (with a chosen directory for downloading the agent) and your organization's instance of Contrast

Start the .NET Framework agent wizard

1. Select **Add New** at the top of the Contrast web interface.
2. Select the **Application** card.
3. Under Select a language, select **.NET Framework**.
4. Select the operating system you are using.

Select an application

Locate an ASP.NET Framework web application where you can add the Contrast extension.

If you do not have a web app, create an application hosted on Azure App Service. Then, publish your application to Azure and confirm that it works as expected without Contrast.

Configure application settings

Configure the displayed settings to let the agent connect to Contrast. Add the displayed values in the Application Settings section of the Environment variables blade for your application.

Add extension for the application

1. In the [Azure Portal](#), select your hosted application.
2. Select **Extensions**.
3. Select **Add**.

4. Select **Contrast .NET Framework Site Extension for Azure App Service**.
5. Select **OK** and agree to the Terms and Conditions.
6. Wait a few seconds and confirm the site extension is correctly installed.
7. Go back to the application overview and **Restart** the application.
8. Go to the application, and confirm the application is reporting to Contrast.

Verify the agent deployment

1. Interact with your application as you normally would. For example, interact with your application's web interface or send API commands
2. Go to the Contrast web interface and select **Applications** in the header.
3. Verify that you see the name of your application in the list.
4. Select servers in the header.
5. Verify that you see the name of your local server in the list.

.NET Framework installation and configuration in a container

Use this workflow with the Add New .NET Framework agent wizard to ensure you have all the steps covered for installing and configuring the .NET Framework agent in a container. Using the wizard is the simplest way to deploy a new agent.

Before you begin

Make sure you have everything you need before you start.

- The agent will need to be able to reach your Contrast instance. It can be a local/on-premise instance or a hosted instance. A proxy can be configured if the environment has limited network access.
- It must use [supported technologies \(page 212\)](#) and [system requirements \(page 213\)](#)
- Understand the [order of precedence \(page 106\)](#)
- You will also need access to a command line interface (with a chosen directory for downloading the agent) and your organization's instance of Contrast

Start the .NET Framework agent wizard

1. Select **Add New** at the top of the Contrast web interface.
2. Select the **Application** card.
3. Under Select a language, select **.NET Framework**.
4. Select the operating system you are using.

Install the image with the agent

1. Under Select application deployment method, select **Install with a container**.
2. Under Select method to deploy with containers, select **Dockerfile**.
3. Under Build image with agent, copy the displayed commands and add them to your Dockerfile.
4. Under Build image with agent, copy the displayed command to build the image.

Configure the agent

Under Configure the agent, select the best options to get the configuration details that the agent needs to communicate with Contrast:



NOTE

You have the option of using the [Contrast Agent Configuration Editor \(page 109\)](#) to configure agent settings. Select **Use configuration editor** next to Use connection token or Use API configuration.

1. Select the agent key that you want to use.

If only one key exists, it's preselected.

[Configure agent keys \(page 1162\)](#) describes how to create multiple agent keys.

2. Optionally, select **Register the application**.
When you register an application, you can assign it to groups for access control. You can also assign tags and metadata to the application.
3. For agent versions 51.0.40 and later, select **Use Connection Token**.
This configuration method uses a single variable to specify authentication values. The Connection Token is the recommended configuration option.
Copy the displayed code and set the variable in a location where the agent has access to it.
4. For agent versions earlier than 51.0.40, select **Use API configuration**.
This configuration method uses individual variables to specify authentication values.
Copy the displayed code and set these variables in a location where the agent has access to them.
5. If you prefer to configure the agent using a YAML file, select **Download configuration**.
Copy the displayed code to generate a YAML file that contains your authentication settings.

Add Contrast configuration to the image

Under Add Contrast configuration to the image, add the displayed command to your Docker file to download the agent configuration (YAML) file to the same location as your Dockerfile.

Run your container

Copy the displayed command to run your container.

Verify the agent deployment

1. Interact with your application as you normally would. For example, interact with your application's web interface or send API commands
2. Go to the Contrast web interface and select **Applications** in the header.
3. Verify that you see the name of your application in the list.
4. Select servers in the header.
5. Verify that you see the name of your local server in the list.

.NET Framework installation and configuration workflow with pipelines

Use this workflow with the Add New .NET Framework agent wizard to ensure you have all the steps covered for installing and configuring the .NET Framework agent in a pipeline. Using the wizard is the simplest way to deploy a new agent.

Before you begin

Make sure you have everything you need before you start.

- The agent will need to be able to reach your Contrast instance. It can be a local/on-premise instance or a hosted instance. A proxy can be configured if the environment has limited network access.
- It must use [supported versions, frameworks, and tools \(page 120\)](#)
- Understand the [order of precedence \(page 106\)](#)
- You will also need access to a command line interface (with a chosen directory for downloading the agent) and your organization's instance of Contrast

Start the .NET Framework agent wizard

1. Select **Add New** at the top of the Contrast web interface.
2. Select the **Application** card.

3. Under Select a language, select **.NET Framework**.
4. Select the operating system you are using.

Install the agent

1. Under Select a method to deploy an agent, select **Install with a pipeline**.
2. Under Select method to deploy with pipelines, select **GitHub** or **Bitbucket**.
3. Under Install agent, copy the displayed commands to add a step to a GitHub action or a Bitbucket pipeline.
4. Under Install an agent, copy the displayed commands to add the step running the application.

Configure the agent

Under Configure the agent, select the best options to get the configuration details that the agent needs to communicate with Contrast:



NOTE

You have the option of using the [Contrast Agent Configuration Editor \(page 109\)](#) to configure agent settings. Select **Use configuration editor** next to Use connection token or Use API configuration.

1. Select the agent key that you want to use.

If only one key exists, it's preselected.

[Configure agent keys \(page 1162\)](#) describes how to create multiple agent keys.

2. Optionally, select **Register the application**.
When you register an application, you can assign it to groups for access control. You can also assign tags and metadata to the application.
3. For agent versions 51.0.40 and later, select **Use Connection Token**.
This configuration method uses a single variable to specify authentication values. The Connection Token is the recommended configuration option.
Copy the displayed code and set the variable in a location where the agent has access to it.
4. For agent versions earlier than 51.0.40, select **Use API configuration**.
This configuration method uses individual variables to specify authentication values.
Copy the displayed code and set these variables in a location where the agent has access to them.
5. If you prefer to configure the agent using a YAML file, select **Download configuration**.
Copy the displayed code to generate a YAML file that contains your authentication settings.

Verify the agent deployment

1. Interact with your application as you normally would. For example, interact with your application's web interface or send API commands
2. Go to the Contrast web interface and select **Applications** in the header.
3. Verify that you see the name of your application in the list.
4. Select servers in the header.
5. Verify that you see the name of your local server in the list.

.NET Core installation and configuration workflows

Follow these workflows to make sure you have covered the steps for installing and configuring the .NET Core agent.

- [.NET Core basic install \(page 68\)](#)
- [.NET Core with an installer \(page 69\)](#)
- [.NET Core with Azure App Service \(page 70\)](#)
- [.NET Core in a container \(page 71\)](#)
- [.NET Core in a pipeline \(page 73\)](#)
- [.NET Core with an infrastructure as code tool \(Ansible playbook\) \(page 94\)](#)

.NET Core basic installation and configuration

Use this workflow with the Add New .NET Core agent wizard to ensure you have all the steps covered for installing and configuring the .NET Core agent with a basic installation. Using the wizard is the simplest way to deploy a new agent.

Before you begin

Make sure you have everything you need before you start.

- The agent will need to be able to reach your Contrast instance. It can be a local/on-premise instance or a hosted instance. A proxy can be configured if the environment has limited network access.
- It must use [supported technologies \(page 271\)](#) and [system requirements \(page 272\)](#)
- Understand the [order of precedence \(page 106\)](#)
- You will also need access to a command line interface (with a chosen directory for downloading the agent) and your organization's instance of Contrast

Start the .NET Core agent wizard

1. Select **Add New** at the top of the Contrast web interface.
2. Select the **Application** card.
3. Under Select a language, select **.NET Core**.
4. Select the operating system you are using.

Install the agent

1. Under Select application deployment method, select **Install manually**.
2. Under Install agent, copy the displayed commands to complete the installation.

Configure the agent

Under Configure the agent, select the best options to get the configuration details that the agent needs to communicate with Contrast:



NOTE

You have the option of using the [Contrast Agent Configuration Editor \(page 109\)](#) to configure agent settings. Select **Use configuration editor** next to Use connection token or Use API configuration.

1. Select the agent key that you want to use.

If only one key exists, it's preselected.

[Configure agent keys \(page 1162\)](#) describes how to create multiple agent keys.

2. Optionally, select **Register the application**.

When you register an application, you can assign it to groups for access control. You can also assign tags and metadata to the application.

3. For agent versions 4.2.22 and later, select **Use Connection Token**.
This configuration method uses a single variable to specify authentication values. The Connection Token is the recommended configuration option.
Copy the displayed code and set the variable in a location where the agent has access to it.
4. For agent versions earlier than 4.2.22, select **Use API configuration**.
This configuration method uses individual variables to specify authentication values.
Copy the displayed code and set these variables in a location where the agent has access to them.
5. If you prefer to configure the agent using a YAML file, select **Download configuration**.
Copy the displayed code to generate a YAML file that contains your authentication settings.

Set environment variables

Under Set environment variables, set the displayed environment values for your application's process.

Run Contrast with your application

Under Run Contrast, copy the displayed command to run your application.

Verify the agent deployment

1. Interact with your application as you normally would. For example, interact with your application's web interface or send API commands
2. Go to the Contrast web interface and select **Applications** in the header.
3. Verify that you see the name of your application in the list.
4. Select servers in the header.
5. Verify that you see the name of your local server in the list.

.NET Core installation and configuration with an installer

Use this workflow with the Add New .NET Core agent wizard to ensure you have all the steps covered for installing and configuring the .NET Core agent with an installer. Using the wizard is the simplest way to deploy a new agent.

Before you begin

Make sure you have everything you need before you start.

- The agent will need to be able to reach your Contrast instance. It can be a local/on-premise instance or a hosted instance. A proxy can be configured if the environment has limited network access.
- It must use [supported technologies \(page 271\)](#) and [system requirements \(page 272\)](#)
- Understand the [order of precedence \(page 106\)](#)
- You will also need access to a command line interface (with a chosen directory for downloading the agent) and your organization's instance of Contrast

Start the .NET Core agent wizard

1. Select **Add New** at the top of the Contrast web interface.
2. Select the **Application** card.
3. Under Select a language, select **.NET Core**.
4. Select the **Windows** operating system.

Download the installer

1. Under Select application deployment method, select **Install with Contrast Windows installer**.
2. Under Download the installer, copy the displayed commands to download the Contrast Windows installer.

Configure the agent

Under Configure the agent, select the best options to get the configuration details that the agent needs to communicate with Contrast:



NOTE

You have the option of using the [Contrast Agent Configuration Editor \(page 109\)](#) to configure agent settings. Select **Use configuration editor** next to Use connection token or Use API configuration.

1. Select the agent key that you want to use.

If only one key exists, it's preselected.

[Configure agent keys \(page 1162\)](#) describes how to create multiple agent keys.

2. Optionally, select **Register the application**.
When you register an application, you can assign it to groups for access control. You can also assign tags and metadata to the application.
3. For agent versions 4.2.22 and later, select **Use Connection Token**.
This configuration method uses a single variable to specify authentication values. The Connection Token is the recommended configuration option.
Copy the displayed code and set the variable in a location where the agent has access to it.
4. For agent versions earlier than 4.2.22, select **Use API configuration**.
This configuration method uses individual variables to specify authentication values.
Copy the displayed code and set these variables in a location where the agent has access to them.
5. If you prefer to configure the agent using a YAML file, select **Download configuration**.
Copy the displayed code to generate a YAML file that contains your authentication settings.

Run the installer

1. Extract the downloaded ZIP archive on the web server.
2. Under Run installer, use the displayed command to run `contrast-dotnet-core-agent-for-iis-installer.exe` from the content directory located in the extracted directory.

Verify the agent deployment

1. Interact with your application as you normally would. For example, interact with your application's web interface or send API commands
2. Go to the Contrast web interface and select **Applications** in the header.
3. Verify that you see the name of your application in the list.
4. Select servers in the header.
5. Verify that you see the name of your local server in the list.

.NET Core installation and configuration with Azure App Service

Use this workflow with the Add New .NET Core agent wizard to ensure you have all the steps covered for installing and configuring the .NET Core agent with the Azure App Service. Using the wizard is the simplest way to deploy a new agent.

Before you begin

Make sure you have everything you need before you start.

- The agent will need to be able to reach your Contrast instance. It can be a local/on-premise instance or a hosted instance. A proxy can be configured if the environment has limited network access.
- It must use [supported technologies \(page 271\)](#) and [system requirements \(page 272\)](#)
- Understand the [order of precedence \(page 106\)](#)
- You will also need access to a command line interface (with a chosen directory for downloading the agent) and your organization's instance of Contrast

Start the .NET Core agent wizard

1. Select **Add New** at the top of the Contrast web interface.
2. Select the **Application** card.
3. Under Select a language, select **.NET Core**.
4. Select **Windows** for operating system.

Install the agent

1. Under Select application deployment method, select **Install with Azure**.
2. Locate an ASP.NET Core web app where you can add the Contrast extension.
If you do not have a web app, [create an application](#) hosted on Azure App Service. Then publish your application to Azure, and confirm that it works as expected without Contrast.

Configure application settings

Under Configure application settings, use the displayed values to configure settings that allow the agent to connect to Contrast. Add these values in the Application Settings section of the Environment variables blade for your application.

Add an extension for the application

1. Go to the [Azure Portal](#).
2. Select your hosted application.
3. Select **Extensions**.
4. Select **Add**.
5. Select **Contrast .NET Core Site Extension for Azure App Service** (this is the extension for .NET Core applications).
6. Select **OK** and agree to the terms and conditions.
7. Wait a few seconds and confirm the site extension installed correctly.
8. Go back to the application overview and **Restart** the application.

Verify the agent deployment

1. Interact with your application as you normally would. For example, interact with your application's web interface or send API commands
2. Go to the Contrast web interface and select **Applications** in the header.
3. Verify that you see the name of your application in the list.
4. Select servers in the header.
5. Verify that you see the name of your local server in the list.

.NET Core installation and configuration in a container

Use this workflow with the Add New .NET Core agent wizard to ensure you have all the steps covered for installing and configuring the .NET Core agent with a container. Using the wizard is the simplest way to deploy a new agent.

Before you begin

Make sure you have everything you need before you start.

- The agent will need to be able to reach your Contrast instance. It can be a local/on-premise instance or a hosted instance. A proxy can be configured if the environment has limited network access.
- It must use [supported technologies \(page 271\)](#) and [system requirements \(page 272\)](#)
- Understand the [order of precedence \(page 106\)](#)
- You will also need access to a command line interface (with a chosen directory for downloading the agent) and your organization's instance of Contrast

Start the .NET Core agent wizard

1. Select **Add New** at the top of the Contrast web interface.
2. Select the **Application** card.
3. Under Select a language, select **.NET Core**.
4. Select the operating system you are using.

Select method to deploy with containers

1. Under Select application deployment method, select **Install in a container**.
2. Under Select method, to deploy containers, select **Dockerfile** or **Kubernetes** (Linux only).

Dockerfile: Build image with agent

1. Under Build image with agent, copy the displayed commands and add them to your Dockerfile.
2. Use the displayed command to build the image.

Dockerfile: Configure the agent

Under Configure the agent, select the best options to get the configuration details that the agent needs to communicate with Contrast:



NOTE

You have the option of using the [Contrast Agent Configuration Editor \(page 109\)](#) to configure agent settings. Select **Use configuration editor** next to Use connection token or Use API configuration.

1. Select the agent key that you want to use.

If only one key exists, it's preselected.

[Configure agent keys \(page 1162\)](#) describes how to create multiple agent keys.

2. Optionally, select **Register the application**.
When you register an application, you can assign it to groups for access control. You can also assign tags and metadata to the application.
3. For agent versions 4.2.22 and later, select **Use Connection Token**.
This configuration method uses a single variable to specify authentication values. The Connection Token is the recommended configuration option.
Copy the displayed code and set the variable in a location where the agent has access to it.
4. For agent versions earlier than 4.2.22, select **Use API configuration**.
This configuration method uses individual variables to specify authentication values.
Copy the displayed code and set these variables in a location where the agent has access to them.
5. If you prefer to configure the agent using a YAML file, select **Download configuration**.
Copy the displayed code to generate a YAML file that contains your authentication settings.

Dockerfile: Add Contrast configuration to image

Under Add Contrast configuration to image, add the displayed command to your Dockerfile.

This command downloads the agent the configuration (YAML) file to the same location as your Dockerfile. Use this file to configure settings that you want to use across all instances of the application

Dockerfile: Run your container

Under Run your container, copy the displayed command to run the container.

Kubernetes: (Linux only)

1. Under Select method to configure agent operator, select **Helm** (recommended) or **Manifest**.
2. **For Helm:**
 1. Under Download and set up files, copy the displayed commands to create a values files.
 2. Under Label the deployments, use the displayed commands to label your deployments.
 3. Under Install and deploy Helm chart, copy the displayed commands to complete the configuration.
3. **For Manifest:**
 1. Under Install operator, copy the displayed commands to install the Contrast agent operator.
 2. Under Configure operator, copy the displayed commands to configure the Contrast agent operator.
 3. Under Inject workloads, copy the displayed commands to configure an AgentInjector entity.
 4. Optionally, under Configure cluster agent and cluster, copy the displayed commands to configure these entities.

Verify the agent deployment

1. Interact with your application as you normally would. For example, interact with your application's web interface or send API commands
2. Go to the Contrast web interface and select **Applications** in the header.
3. Verify that you see the name of your application in the list.
4. Select servers in the header.
5. Verify that you see the name of your local server in the list.

.NET Core installation and configuration in a pipeline

Use this workflow with the Add New .NET Core agent wizard to ensure you have all the steps covered for installing and configuring the .NET Core agent in a pipeline. Using the wizard is the simplest way to deploy a new agent.

Before you begin

Make sure you have everything you need before you start.

- The agent will need to be able to reach your Contrast instance. It can be a local/on-premise instance or a hosted instance. A proxy can be configured if the environment has limited network access.
- It must use [supported technologies \(page 271\)](#) and [system requirements \(page 272\)](#)
- Understand the [order of precedence \(page 106\)](#)
- You will also need access to a command line interface (with a chosen directory for downloading the agent) and your organization's instance of Contrast

Start the .NET Core agent wizard

1. Select **Add New** at the top of the Contrast web interface.
2. Select the **Application** card.

3. Under Select a language, select **.NET Core**.
4. Select the operating system you are using.

Install the agent

1. Under Select method to deploy with pipelines select **GitHub** or **Bitbucket**.
2. Under Install the agent, copy the displayed commands to add a step to your GitHub action or your Bitbucket pipeline.
3. Under Install the agent, copy the displayed command to add variables to the step running your application

Configure the agent

Under Configure the agent, select the best options to get the configuration details that the agent needs to communicate with Contrast:



NOTE

You have the option of using the [Contrast Agent Configuration Editor \(page 109\)](#) to configure agent settings. Select **Use configuration editor** next to Use connection token or Use API configuration.

1. Select the agent key that you want to use.

If only one key exists, it's preselected.

[Configure agent keys \(page 1162\)](#) describes how to create multiple agent keys.

2. Optionally, select **Register the application**.
When you register an application, you can assign it to groups for access control. You can also assign tags and metadata to the application.
3. Select **Use Connection Token**.
This configuration method uses a single variable to specify authentication values. The Connection Token is the recommended configuration option.
Copy the displayed code and set the variable in a location where the agent has access to it.

Verify the agent deployment

1. Interact with your application as you normally would. For example, interact with your application's web interface or send API commands
2. Go to the Contrast web interface and select **Applications** in the header.
3. Verify that you see the name of your application in the list.
4. Select servers in the header.
5. Verify that you see the name of your local server in the list.

Node.js installation and configuration workflows

Follow these workflows to make sure you have covered the steps for installing and configuring the Node.js agent.

- [Node.js basic install \(page 75\)](#)
- [Node.js in a container \(page 76\)](#)
- [Node.js with a pipeline \(page 78\)](#)

- [Node.js with Cloud deployment integrations \(page 79\)](#)
- [Ansible playbook for Contrast \(page 94\)](#)

Node.js basic installation and configuration

Use this workflow with the Add New Node.js agent wizard to ensure you have all the steps covered for installing and configuring the Node.js with the basic installation. Using the wizard is the simplest way to deploy a new agent.

Before you begin

Make sure you have everything you need before you start.

- The agent will need to be able to reach your Contrast instance. It can be a local/on-premise instance or a hosted instance. A proxy can be configured if the environment has limited network access.
- It must use [supported technologies \(page 332\)](#) and [system requirements \(page 333\)](#)
- Understand the [order of precedence \(page 106\)](#)
- You will also need access to a command line interface (with a chosen directory for downloading the agent) and your organization's instance of Contrast

Start the Node.js agent wizard

1. Select **Add New** at the top of the Contrast web interface.
2. Select the **Application** card.
3. Under Select a language, select **Node**.
4. Select the operating system you are using.

Install the agent

1. Under Select application deployment method, select **Install manually**.
2. Under Install agent, select any of these options:
 - Install the latest version of the agent from [npm](#)
 - Run the yarn command
3. Add the provided command to the scripts section of your application's `package.json` file.

Configure the agent

Under Configure the agent, select the best options to get the configuration details that the agent needs to communicate with Contrast:



NOTE

You have the option of using the [Contrast Agent Configuration Editor \(page 109\)](#) to configure agent settings. Select **Use configuration editor** next to Use connection token or Use API configuration.

1. Select the agent key that you want to use.

If only one key exists, it's preselected.

[Configure agent keys \(page 1162\)](#) describes how to create multiple agent keys.

2. Optionally, select **Register the application**.

When you register an application, you can assign it to groups for access control. You can also assign tags and metadata to the application.

3. For agent versions 5.15.0 and later, select **Use Connection Token**.
This configuration method uses a single variable to specify authentication values.
Copy the displayed code and set the variable in a location where the agent has access to it.
4. For agent versions earlier than 5.15.0, select **Use API configuration**.
This configuration method uses individual variables to specify authentication values.
Copy the displayed code and set these variables in a location where the agent has access to them.
5. If you prefer to configure the agent using a YAML file, select **Download configuration**.
Copy the displayed code to generate a YAML file that contains your authentication settings.
6. Run your application with the agent using the displayed code

Verify the agent deployment

1. Interact with your application as you normally would. For example, interact with your application's web interface or send API commands.
2. Go to the Contrast web interface and select **Applications** in the header.
3. Verify that you see the name of your application in the list.
4. Select servers in the header.
5. Verify that you see the name of your local server in the list.

Node.js installation and configuration in a container

Use this workflow with the Add New Node.js agent wizard to ensure you have all the steps covered for installing and configuring the Node.js in a container. Using the wizard is the simplest way to deploy a new agent.

Before you begin

Make sure you have everything you need before you start.

- The agent will need to be able to reach your Contrast instance. It can be a local/on-premise instance or a hosted instance. A proxy can be configured if the environment has limited network access.
- It must use [supported technologies \(page 332\)](#) and [system requirements \(page 333\)](#)
- Understand the [order of precedence \(page 106\)](#)
- You will also need access to a command line interface (with a chosen directory for downloading the agent) and your organization's instance of Contrast

Start the Node.js agent wizard

1. Select **Add New** at the top of the Contrast web interface.
2. Select the **Application** card.
3. Under Select a language, select **Node**.
4. Select the operating system you are using.

Select method to deploy with containers

1. Under Select application deployment method, select **Install with a container**.
2. Under Select method to deploy with containers, select Dockerfile or Kubernetes (Linux or OS X only) or follow agent installation instructions (for Windows).

Install with Linux or OS X

Select method to deploy with containers

Under Select method to deploy with containers, select **Dockerfile** or **Kubernetes**.

Dockerfile: Build image with the agent

1. Under Build image with agent, copy the displayed command and add it to your Dockerfile.
2. You can modify your application with the provided command to include the Agent.
3. Copy the displayed command to build the image.

Dockerfile: Configure the agent

Under Configure the agent, select the best options to get the configuration details that the agent needs to communicate with Contrast:



NOTE

You have the option of using the [Contrast Agent Configuration Editor \(page 109\)](#) to configure agent settings. Select **Use configuration editor** next to Use connection token or Use API configuration.

1. Select the agent key that you want to use.

If only one key exists, it's preselected.

[Configure agent keys \(page 1162\)](#) describes how to create multiple agent keys.

2. Optionally, select **Register the application**.
When you register an application, you can assign it to groups for access control. You can also assign tags and metadata to the application.
3. For agent versions 5.15.0 and later, select **Use Connection Token**.
This configuration method uses a single variable to specify authentication values.
Copy the displayed code and set the variable in a location where the agent has access to it.
4. For agent versions earlier than 5.15.0, select **Use API configuration**.
This configuration method uses individual variables to specify authentication values.
Copy the displayed code and set these variables in a location where the agent has access to them.
5. If you prefer to configure the agent using a YAML file, select **Download configuration**.
Copy the displayed code to generate a YAML file that contains your authentication settings.
6. Continue by verifying the agent deployment.

Kubernetes: Select a method to configure agent operator

1. Under Select method to configure agent operator, select **Helm** (recommended) or **Manifest**.
2. **For Helm:**
 1. Under Download and set up files, copy the displayed commands to create a values files.
 2. Under Label the deployments, use the displayed commands to label your deployments.
 3. Under the Install and deploy Helm chart, copy the displayed commands to complete the configuration.
3. **For Manifest:**
 1. Under Install operator, copy the displayed commands to install the Contrast agent operator.
 2. Under Configure operator, copy the displayed commands to configure the Contrast agent operator.
 3. Under Inject workloads, copy the displayed commands to configure an AgentInjector entity.
 4. Optionally, under Configure cluster agent and agent, copy the displayed commands to configure these entities.

4. Continue by verifying the agent deployment.

Install with Windows

Build image with the agent

1. Under Build image with agent, copy the displayed command and add it to your Dockerfile.
2. You can modify your application with the provided command to include the Agent.
3. Copy the displayed command to build the image.

Configure the agent

Under Configure the agent, select the best options to get the configuration details that the agent needs to communicate with Contrast:

1. Optionally, select **Register the application**.
When you register an application, you can assign it to groups for access control. You can also assign tags and metadata to the application.
2. For agent versions 5.15.0 and later, select **Use Connection Token**.
This configuration method uses a single variable to specify authentication values.
Copy the displayed code and set the variable in a location where the agent has access to it.
3. For agent versions earlier than 5.15.0, select **Use API configuration**.
This configuration method uses individual variables to specify authentication values.
Copy the displayed code and set these variables in a location where the agent has access to them.
4. If you prefer to configure the agent using a YAML file, select **Download configuration**.
Copy the displayed code to generate a YAML file that contains your authentication settings.
5. Continue by verifying the agent deployment.

Verify the agent deployment

1. Interact with your application as you normally would. For example, interact with your application's web interface or send API commands.
2. Go to the Contrast web interface and select **Applications** in the header.
3. Verify that you see the name of your application in the list.
4. Select servers in the header.
5. Verify that you see the name of your local server in the list.

Node.js installation and configuration with a pipeline

Use this workflow with the Add New Node.js agent wizard to ensure you have all the steps covered for installing and configuring the Node.js with a pipeline. Using the wizard is the simplest way to deploy a new agent.

Before you begin

Make sure you have everything you need before you start

- The agent will need to be able to reach your Contrast instance. It can be a local/on-premise instance or a hosted instance. A proxy can be configured if the environment has limited network access.
- It must use [supported technologies \(page 332\)](#) and [system requirements \(page 333\)](#)
- Understand the [order of precedence \(page 106\)](#)
- You will also need access to a command line interface (with a chosen directory for downloading the agent) and your organization's instance of Contrast

Start the Node.js agent wizard

1. Select **Add New** at the top of the Contrast web interface.

2. Select the **Application** card.
3. Under Select a language, select **Node**.
4. Select the operating system you are using.

Install the agent

1. Under Select application deployment method, select **Install with a pipeline**.
2. Under Select method to deploy pipelines, select **Github** or **Bitbucket**.
3. Under Install Agent, copy the displayed commands to install the agent.

Configure the agent

Under Configure the agent, select the best options to get the configuration details that the agent needs to communicate with Contrast:



NOTE

You have the option of using the [Contrast Agent Configuration Editor \(page 109\)](#) to configure agent settings. Select **Use configuration editor** next to Use connection token or Use API configuration.

1. Select the agent key that you want to use.

If only one key exists, it's preselected.

[Configure agent keys \(page 1162\)](#) describes how to create multiple agent keys.

2. Optionally, select **Register the application**.
When you register an application, you can assign it to groups for access control. You can also assign tags and metadata to the application.
3. Select **Use Connection Token**.
This configuration method uses a single variable to specify authentication values. The Connection Token is the recommended configuration option.
Copy the displayed code and set the variable in a location where the agent has access to it.

Verify the agent deployment

1. Interact with your application as you normally would. For example, interact with your application's web interface or send API commands.
2. Go to the Contrast web interface and select **Applications** in the header.
3. Verify that you see the name of your application in the list.
4. Select servers in the header.
5. Verify that you see the name of your local server in the list.

Node.js installation and configuration with Cloud deployment integrations

Use this workflow to ensure you have all the steps covered for installing and configuring Node.js with Cloud deployment integrations.

Before you begin

Make sure you have everything you need before you start.

- The agent will need to be able to reach your Contrast instance. It can be a local/on-premise instance or a hosted instance. A proxy can be configured if the environment has limited network access.

- It must use [supported technologies \(page 332\)](#) and [system requirements \(page 333\)](#)
- Understand the [order of precedence \(page 106\)](#)
- You will also need access to a command line interface (with a chosen directory for downloading the agent) and your organization's instance of Contrast

Steps

1. **Set up variables.** Configure agent authentication configuration variables.
 - Set the minimum configuration values defined [here \(page 102\)](#)
 - Configure additional values (application metadata, session metadata)
2. **Continue with deployment type.** Install based on the cloud deployment type.
 - With [IBM Cloud \(page 344\)](#)
 - With [VMware Tanzu](#)
3. **Verify.** To verify that Contrast is working, use your application as you usually would. For example, click on your application's web interface, or send some API commands.
Then in the Contrast web interface, select **Applications** in the header. You should see the name of your application.
You can also select **Server** in the header and you should see the hostname of your (local) server.

PHP installation and configuration workflow

Follow this workflow to make sure you have covered the steps for installing and configuring the PHP agent.

- [PHP by repository \(manually\) \(page 80\)](#)
- [PHP in a container \(page 81\)](#)
- [PHP with a pipeline \(page 83\)](#)

PHP installation and configuration by repository (manually)

Use this workflow with Add New PHP agent wizard to ensure you have all the steps covered for installing and configuring PHP with manual installation. Using the wizard is the simplest way to deploy a new agent.

Before you begin

Make sure you have everything you need before you start.

- The agent will need to be able to reach your Contrast instance. It can be a local/on-premise instance or a hosted instance. A proxy can be configured if the environment has limited network access.
- It must use [supported technologies \(page 391\)](#) and [system requirements \(page 392\)](#)
- Understand the [order of precedence \(page 106\)](#)
- You will also need access to a command line interface (with a chosen directory for downloading the agent) and your organization's instance of Contrast

Start the PHP agent wizard

1. Select **Add New** at the top of the Contrast web interface.
2. Select the **Application** card.
3. Under Select a language, select **PHP**.
4. Select the operating system you are using.

Install the agent

1. Under Select application deployment method, select **Install manually**.
2. Under the Installation method, select **Debian** or **RPM**.

3. Under Install and configure, install the agent package using the provided commands to register it into your system.
4. Use the provided command lines to configure and enable the agent in your environment.

Configure the agent

Under Configure the agent, select the best options to get the configuration details that the agent needs to communicate with Contrast:



NOTE

You have the option of using the [Contrast Agent Configuration Editor \(page 109\)](#) to configure agent settings. Select **Use configuration editor** next to Use connection token or Use API configuration.

1. Select the agent key that you want to use.

If only one key exists, it's preselected.

[Configure agent keys \(page 1162\)](#) describes how to create multiple agent keys.

2. Optionally, select **Register the application**.
When you register an application, you can assign it to groups for access control. You can also assign tags and metadata to the application.
3. For agent versions 1.34.0 and later, select **Use Connection Token**.
This configuration method uses a single variable to specify authentication values.
Copy the displayed code and set the variable in a location where the agent has access to it.
4. For agent versions earlier than 1.34.0, select **Use API configuration**.
This configuration method uses individual variables to specify authentication values.
Copy the displayed code and set these variables in a location where the agent has access to them.
5. If you prefer to configure the agent using a YAML file, select **Download configuration**.
Copy the displayed code to generate a YAML file that contains your authentication settings.

Verify the agent deployment

1. Interact with your application as you normally would. For example, interact with your application's web interface or send API commands.
2. Go to the Contrast web interface and select **Applications** in the header.
3. Verify that you see the name of your application in the list.
4. Select servers in the header.
5. Verify that you see the name of your local server in the list.

PHP installation and configuration in a container

Use this workflow with the Add New PHP agent wizard to ensure you have all the steps covered for installing and configuring the PHP agent in a container. Using the wizard is the simplest way to deploy a new agent.

Before you begin

Make sure you have everything you need before you start.

- The agent will need to be able to reach your Contrast instance. It can be a local/on-premise instance or a hosted instance. A proxy can be configured if the environment has limited network access.

- It must use [supported versions, frameworks, and tools \(page 120\)](#)
- Understand the [order of precedence \(page 106\)](#)
- You will also need access to a command line interface (with a chosen directory for downloading the agent) and your organization's instance of Contrast

Start the PHP agent wizard

1. Select **Add New** at the top of the Contrast web interface.
2. Select the **Application** card.
3. Under Select a language, select **PHP**.
4. Select the operating system you are using.

Select method to deploy with containers

1. Under Select application deployment method, select **Install with a container**.
2. Under Select method to deploy with containers, select **Dockerfile** or **Kubernetes**.

Dockerfile: Build image with the agent

1. Under Build image with agent, copy the displayed command and add it to your Dockerfile.
2. Copy the displayed command to build the image.

Dockerfile: Configure the agent

Under Configure the agent, select the best options to get the configuration details that the agent needs to communicate with Contrast:



NOTE

You have the option of using the [Contrast Agent Configuration Editor \(page 109\)](#) to configure agent settings. Select **Use configuration editor** next to Use connection token or Use API configuration.

1. Select the agent key that you want to use.

If only one key exists, it's preselected.

[Configure agent keys \(page 1162\)](#) describes how to create multiple agent keys.

2. Optionally, select **Register the application**.
When you register an application, you can assign it to groups for access control. You can also assign tags and metadata to the application.
3. For agent versions 1.34.0 and later, select **Use Connection Token**.
This configuration method uses a single variable to specify authentication values. The Connection Token is the recommended configuration option.
Copy the displayed code and set the variable in a location where the agent has access to it.
4. For agent versions earlier than 1.34.0, select **Use API configuration**.
This configuration method uses individual variables to specify authentication values.
Copy the displayed code and set these variables in a location where the agent has access to them.
5. If you prefer to configure the agent using a YAML file, select **Download configuration**.
Copy the displayed code to generate a YAML file that contains your authentication settings.

Kubernetes

1. Under Select method to configure agent operator, select **Helm charts** (recommended) or **Manifest**.
2. **For Helm:**
 1. Under Download and set up files, copy the displayed commands to create a values file.
 2. Under Label the deployments, use the displayed commands to label your deployments.
 3. Under Install and deploy Helm chart, copy the displayed commands to complete the configuration.
3. **For Manifest:**
 1. Under Install operator, copy the displayed commands to install the Contrast agent operator.
 2. Under Configure operator, copy the displayed commands to configure the Contrast agent operator.
 3. Under Inject workloads, copy the displayed commands to configure an AgentInjector entity.
 4. Optionally, under Configure cluster agent and agent, copy the displayed commands to configure these entities.

Verify the agent deployment

1. Interact with your application as you normally would. For example, interact with your application's web interface or send API commands
2. Go to the Contrast web interface and select **Applications** in the header.
3. Verify that you see the name of your application in the list.
4. Select servers in the header.
5. Verify that you see the name of your local server in the list.

PHP installation and configuration with a pipeline

Use this workflow with the Add New PHP agent wizard to ensure you have all the steps covered for installing and configuring the PHP agent with a pipeline. Using the wizard is the simplest way to deploy a new agent.

Before you begin

Make sure you have everything you need before you start.

- The agent will need to be able to reach your Contrast instance. It can be a local/on-premise instance or a hosted instance. A proxy can be configured if the environment has limited network access.
- It must use [supported versions, frameworks, and tools \(page 120\)](#)
- Understand the [order of precedence \(page 106\)](#)
- You will also need access to a command line interface (with a chosen directory for downloading the agent) and your organization's instance of Contrast

Start the PHP agent wizard

1. Select **Add New** at the top of the Contrast web interface.
2. Select the **Application** card.
3. Under Select a language, select **PHP**.
4. Select the operating system you are using.

Install the agent

1. Under Select application deployment method, select **Install with a pipeline**.
2. Under Select method to deploy pipelines, select **Github** or **Bitbucket**.
3. Under Select installation method, choose **Debian** or **RPM**.

4. Under Install agent, copy the displayed commands to add a step to your GitHub action or Bitbucket pipeline.

Configure the agent

Under Configure the agent, select the best options to get the configuration details that the agent needs to communicate with Contrast:



NOTE

You have the option of using the [Contrast Agent Configuration Editor \(page 109\)](#) to configure agent settings. Select **Use configuration editor** next to Use connection token or Use API configuration.

1. Select the agent key that you want to use.

If only one key exists, it's preselected.

[Configure agent keys \(page 1162\)](#) describes how to create multiple agent keys.

2. Optionally, select **Register the application**.

When you register an application, you can assign it to groups for access control. You can also assign tags and metadata to the application.

3. Select **Use Connection Token**.

This configuration method uses a single variable to specify authentication values. The Connection Token is the recommended configuration option.

Copy the displayed code and set the variable in a location where the agent has access to it.

Verify the agent deployment

1. Interact with your application as you normally would. For example, interact with your application's web interface or send API commands
2. Go to the Contrast web interface and select **Applications** in the header.
3. Verify that you see the name of your application in the list.
4. Select servers in the header.
5. Verify that you see the name of your local server in the list.

Python installation and configuration workflows

Follow these workflows to make sure you have covered the steps for installing and configuring the Python agent.

- [Python by manual installation \(page 84\)](#)
- [Python in a container \(page 86\)](#)
- [Python with a pipeline \(page 87\)](#)
- [Python with Contrast Runner \(page 467\)](#)

Python installation and configuration by manual installation

Use this workflow with the Add New Python agent wizard to ensure you have all the steps covered for manually installing and configuring the Python. Using the wizard is the simplest way to deploy a new agent.

Before you begin

Make sure you have everything you need before you start.

- The agent will need to be able to reach your Contrast instance. It can be a local/on-premise instance or a hosted instance. A proxy can be configured if the environment has limited network access.
- It must use [supported technologies](#). (page 410)
- Understand the [order of precedence](#) (page 106)
- You will also need access to a command line interface (with a chosen directory for downloading the agent) and your organization's instance of Contrast

Start the Python agent wizard

1. Select **Add New** at the top of the Contrast web interface.
2. Select the **Application** card.
3. Under Select a language, select **Python**.
4. Select the operating system you are using.

Install the agent

1. Under Select application deployment method, select **Install manually**.
2. Under Install agent, Install the agent using pip command.

Configure the agent

Under Configure the agent, select the best options to get the configuration details that the agent needs to communicate with Contrast:



NOTE

You have the option of using the [Contrast Agent Configuration Editor \(page 109\)](#) to configure agent settings. Select **Use configuration editor** next to Use connection token or Use API configuration.

1. Select the agent key that you want to use.

If only one key exists, it's preselected.

[Configure agent keys \(page 1162\)](#) describes how to create multiple agent keys.

2. Optionally, select **Register the application**.
When you register an application, you can assign it to groups for access control. You can also assign tags and metadata to the application.
3. For agent versions 8.6.0 and later, select **Use Connection Token**.
This configuration method uses a single variable to specify authentication values.
Copy the displayed code and set the variable in a location where the agent has access to it.
4. For agent versions earlier than 8.6.0, select **Use API configuration**.
This configuration method uses individual variables to specify authentication values.
Copy the displayed code and set these variables in a location where the agent has access to them.
5. If you prefer to configure the agent using a YAML file, select **Download configuration**.
Copy the displayed code to generate a YAML file that contains your authentication settings.
6. Start your application with the Contrast Runner.

Verify the agent deployment

1. Interact with your application as you normally would. For example, interact with your application's web interface or send API commands.\

2. Go to the Contrast web interface and select **Applications** in the header.
3. Verify that you see the name of your application in the list.
4. Select servers in the header.
5. Verify that you see the name of your local server in the list.

Python installation and configuration in a container

Use this workflow with the Add New Python agent wizard to ensure you have all the steps covered for installing and configuring the Python agent in a container. Using the wizard is the simplest way to deploy a new agent.

Before you begin

Make sure you have everything you need before you start.

- The agent will need to be able to reach your Contrast instance. It can be a local/on-premise instance or a hosted instance. A proxy can be configured if the environment has limited network access.
- It must use [supported versions, frameworks, and tools \(page 120\)](#)
- Understand the [order of precedence \(page 106\)](#)
- You will also need access to a command line interface (with a chosen directory for downloading the agent) and your organization's instance of Contrast

Start the Python agent wizard

1. Select **Add New** at the top of the Contrast web interface.
2. Select the **Application** card.
3. Under Select a language, select **Python**.
4. Select the operating system you are using.

Select method to deploy with containers

1. Under Select application deployment method, select **Install with a container**.
2. Under Select method to deploy with containers, select **Dockerfile** or **Kubernetes**.

Dockerfile: Build image with the agent

1. Under Build image with agent, copy the displayed command and add it to your Dockerfile.
2. Copy the displayed commands to build the image.

Dockerfile: Configure the agent

Under Configure the agent, select the best options to get the configuration details that the agent needs to communicate with Contrast:



NOTE

You have the option of using the [Contrast Agent Configuration Editor \(page 109\)](#) to configure agent settings. Select **Use configuration editor** next to Use connection token or Use API configuration.

1. Select the agent key that you want to use.

If only one key exists, it's preselected.

[Configure agent keys \(page 1162\)](#) describes how to create multiple agent keys.

2. Optionally, select **Register the application**.
When you register an application, you can assign it to groups for access control. You can also assign tags and metadata to the application.
3. For agent versions 8.6.0 and later, select **Use Connection Token**.
This configuration method uses a single variable to specify authentication values. The Connection Token is the recommended configuration option.
Copy the displayed code and set the variable in a location where the agent has access to it.
4. For agent versions earlier than 8.6.0, select **Use API configuration**.
This configuration method uses individual variables to specify authentication values.
Copy the displayed code and set these variables in a location where the agent has access to them.
5. If you prefer to configure the agent using a YAML file, select **Download configuration**.
Copy the displayed code to generate a YAML file that contains your authentication settings.

Kubernetes

1. Under Select method to configure agent operator, select **Helm charts** (recommended) or **Manifest**.
2. **For Helm:**
 1. Under Download and set up files, copy the displayed commands to create a values file.
 2. Under Label the deployments, use the displayed commands to label your deployments.
 3. Under Install and deploy Helm chart, copy the displayed commands to complete the configuration.
3. **For Manifest:**
 1. Under Install operator, copy the displayed commands to install the Contrast agent operator.
 2. Under Configure operator, copy the displayed commands to configure the Contrast agent operator.
 3. Under Inject workloads, copy the displayed commands to configure an AgentInjector entity.
 4. Optionally, under Configure cluster agent and agent, copy the displayed commands to configure these entities.

Verify the agent deployment

1. Interact with your application as you normally would. For example, interact with your application's web interface or send API commands
2. Go to the Contrast web interface and select **Applications** in the header.
3. Verify that you see the name of your application in the list.
4. Select servers in the header.
5. Verify that you see the name of your local server in the list.

Python installation and configuration with a pipeline

Use this workflow with the Add New Python agent wizard to ensure you have all the steps covered for installing and configuring the Python agent with a pipeline. Using the wizard is the simplest way to deploy a new agent.

Before you begin

Make sure you have everything you need before you start.

- The agent will need to be able to reach your Contrast instance. It can be a local/on-premise instance or a hosted instance. A proxy can be configured if the environment has limited network access.
- It must use [supported versions, frameworks, and tools \(page 120\)](#)
- Understand the [order of precedence \(page 106\)](#)
- You will also need access to a command line interface (with a chosen directory for downloading the agent) and your organization's instance of Contrast

Start the Python agent wizard

1. Select **Add New** at the top of the Contrast web interface.
2. Select the **Application** card.
3. Under Select a language, select **Python**.
4. Select the operating system you are using.

Install the agent

1. Under Select application deployment method, select **Install with a pipeline**.
2. Under Select agent installation method, select **Github** or **Bitbucket**.
3. Under Install Agent, copy the displayed commands to install the agent.

Configure the agent

Under Configure the agent, select the best options to get the configuration details that the agent needs to communicate with Contrast:



NOTE

You have the option of using the [Contrast Agent Configuration Editor \(page 109\)](#) to configure agent settings. Select **Use configuration editor** next to Use connection token or Use API configuration.

1. Select the agent key that you want to use.

If only one key exists, it's preselected.

[Configure agent keys \(page 1162\)](#) describes how to create multiple agent keys.

2. Optionally, select **Register the application**.
When you register an application, you can assign it to groups for access control. You can also assign tags and metadata to the application.
3. Select **Use Connection Token**.
This configuration method uses a single variable to specify authentication values. The Connection Token is the recommended configuration option.
Copy the displayed code and set the variable in a location where the agent has access to it.

Verify the agent deployment

1. Interact with your application as you normally would. For example, interact with your application's web interface or send API commands
2. Go to the Contrast web interface and select **Applications** in the header.
3. Verify that you see the name of your application in the list.
4. Select servers in the header.
5. Verify that you see the name of your local server in the list.

Ruby installation and configuration workflow



IMPORTANT

The Ruby Agent is not currently being sold to new customers. Please contact our Sales team if you have any questions about our offerings and support for the Ruby language.

Follow this workflow to ensure you have covered installing and configuring the Ruby agent.

- [Ruby by middleware type \(page 89\)](#)

Ruby installation and configuration by middleware



IMPORTANT

The Ruby Agent is not currently being sold to new customers. Please contact our Sales team if you have any questions about our offerings and support for the Ruby language.

Use this workflow with the Add New Ruby agent wizard to ensure you have all the steps covered for installing and configuring the Ruby agent by middleware. Using the wizard is the simplest way to deploy a new agent.

Before you begin

Make sure you have everything you need before you start.

- The agent will need to be able to reach your Contrast instance. It can be a local/on-premise instance or a hosted instance. A proxy can be configured if the environment has limited network access.
- It must use [supported technologies \(page 469\)](#) and [system requirements \(page 470\)](#)
- Understand the [order of precedence \(page 106\)](#)
- You will also need access to a command line interface (with a chosen directory for downloading the agent) and your organization's instance of Contrast

Start the Ruby agent wizard

1. Select **Add New** at the top of the Contrast web interface.
2. Select the **Application** card.
3. Under Select a language, select **Ruby**.
4. Select the operating system you are using.

Install the agent

1. Under Select application deployment method, select **Install manually**.
2. Under Install Ruby agent, copy the displayed command to install the latest version of the agent or a specific version of the agent.
3. Under Select middleware, choose one of the types of middleware options.

If you select Rails, middleware configuration is not required. You can move to the *Configure the agent* step.

If you select Sinatra or Grape, copy the provided commands to configure your application.

Configure the agent

Under Configure the agent, select the best options to get the configuration details that the agent needs to communicate with Contrast:



NOTE

You have the option of using the [Contrast Agent Configuration Editor \(page 109\)](#) to configure agent settings. Select **Use configuration editor** next to Use connection token or Use API configuration.

1. Select the agent key that you want to use.

If only one key exists, it's preselected.

[Configure agent keys \(page 1162\)](#) describes how to create multiple agent keys.

2. Optionally, select **Register the application**.
When you register an application, you can assign it to groups for access control. You can also assign tags and metadata to the application.
3. Select **Use API configuration**.
This configuration method uses individual variables to specify authentication values.
Copy the displayed code and set these variables in a location where the agent has access to them.
4. If you prefer to configure the agent using a YAML file, select **Download configuration**.
Copy the displayed code to generate a YAML file that contains your authentication settings.

Verify the agent deployment

1. Interact with your application as you normally would. For example, interact with your application's web interface or send API commands.
2. Go to the Contrast web interface and select **Applications** in the header.
3. Verify that you see the name of your application in the list.
4. Select servers in the header.
5. Verify that you see the name of your local server in the list.

Go installation and configuration workflow

Follow this workflow to make sure you have covered the steps for installing and configuring the Go agent.

- [Go with an installer \(page 90\)](#)
- [Go in a container \(page 92\)](#)
- [Go in a pipeline \(page 93\)](#)

Go installation and configuration with an installer

Use this workflow with the Add New Go agent wizard to ensure you have all the steps covered for installing and configuring the Go agent with the Go installer. Using the wizard is the simplest way to deploy a new agent.

Before you begin

Make sure you have everything you need before you start.

- The agent will need to be able to reach your Contrast instance. It can be a local/on-premise instance or a hosted instance. A proxy can be configured if the environment has limited network access.
- It must use [supported technologies \(page 530\)](#)
- You will also need access to a command line interface (with a chosen directory for downloading the agent) and your organization's instance of Contrast

Start the Go agent wizard

1. Select **Add New** at the top of the Contrast web interface.
2. Select the **Application** card.
3. Under Select a language, select **Go**.
4. Select the operating system you are using.

Install the agent

1. Under Select application deployment method, select **Install manually**.
2. Under Run installer, copy the displayed command to install the latest version of the agent or a specific version of the agent.
3. Under Build application, copy the displayed command to build the application.

Configure the agent

Under Configure the agent, select the best options to get the configuration details that the agent needs to communicate with Contrast:



NOTE

You have the option of using the [Contrast Agent Configuration Editor \(page 109\)](#) to configure agent settings. Select **Use configuration editor** next to Use connection token or Use API configuration.

1. Select the agent key that you want to use.

If only one key exists, it's preselected.

[Configure agent keys \(page 1162\)](#) describes how to create multiple agent keys.

2. Optionally, select **Register the application**.
When you register an application, you can assign it to groups for access control. You can also assign tags and metadata to the application.
3. For agent versions 6.11.0 and later, select **Use Connection Token**.
This configuration method uses a single variable to specify authentication values. The Connection Token is the recommended configuration option.
Copy the displayed code and set the variable in a location where the agent has access to it.
4. For agent versions earlier than 6.11.0, select **Use API configuration**.
This configuration method uses individual variables to specify authentication values.
Copy the displayed code and set these variables in a location where the agent has access to them.
5. If you prefer to configure the agent using a YAML file, select **Download configuration**.
Copy the displayed code to generate a YAML file that contains your authentication settings.

Verify the agent deployment

1. Interact with your application as you normally would. For example, interact with your application's web interface or send API commands
2. Go to the Contrast web interface and select **Applications** in the header.
3. Verify that you see the name of your application in the list.
4. Select servers in the header.
5. Verify that you see the name of your local server in the list.

Go agent installation and configuration in a container

Use this workflow with the Add New Go agent wizard to ensure you have all the steps covered for installing and configuring the Go agent with a container. Using the wizard is the simplest way to deploy a new agent.

Before you begin

Make sure you have everything you need before you start.

- The agent will need to be able to reach your Contrast instance. It can be a local/on-premise instance or a hosted instance. A proxy can be configured if the environment has limited network access.
- It must use [supported versions, frameworks, and tools \(page 120\)](#)
- Understand the [order of precedence \(page 106\)](#)
- You will also need access to a command line interface (with a chosen directory for downloading the agent) and your organization's instance of Contrast

Start the Go agent wizard

1. Select **Add New** at the top of the Contrast web interface.
2. Select the **Application** card.
3. Under Select a language, select **Go**.
4. Select the operating system you are using.

Install the agent

1. Under Select application deployment method, select **Install with a container**.
2. Under Select method to deploy with containers, select **Dockerfile**.
3. Under Build the image:
 - a. Copy the displayed command and add it to your Dockerfile.
 - b. Use the displayed command to modify your application command to include the Go agent.
 - c. Use the displayed command to build the image.

Configure the agent

Under Configure the agent, select the best options to get the configuration details that the agent needs to communicate with Contrast:



NOTE

You have the option of using the [Contrast Agent Configuration Editor \(page 109\)](#) to configure agent settings. Select **Use configuration editor** next to Use connection token or Use API configuration.

1. Select the agent key that you want to use.

If only one key exists, it's preselected.

[Configure agent keys \(page 1162\)](#) describes how to create multiple agent keys.

2. Optionally, select **Register the application**.
When you register an application, you can assign it to groups for access control. You can also assign tags and metadata to the application.
3. For agent versions 6.11.0 and later, select **Use Connection Token**.
This configuration method uses a single variable to specify authentication values. The Connection Token is the recommended configuration option.
Copy the displayed code and set the variable in a location where the agent has access to it.
4. For agent versions earlier than 6.11.0, select **Use API configuration**.
This configuration method uses individual variables to specify authentication values.
Copy the displayed code and set these variables in a location where the agent has access to them.
5. If you prefer to configure the agent using a YAML file, select **Download configuration**.
Copy the displayed code to generate a YAML file that contains your authentication settings.

Verify the agent deployment

1. Interact with your application as you normally would. For example, interact with your application's web interface or send API commands
2. Go to the Contrast web interface and select **Applications** in the header.
3. Verify that you see the name of your application in the list.
4. Select servers in the header.
5. Verify that you see the name of your local server in the list.

Go agent installation and configuration in a pipeline

Use this workflow with the Add New Go agent wizard to ensure you have all the steps covered for installing and configuring the Go agent in a pipeline. Using the wizard is the simplest way to deploy a new agent.

Before you begin

Make sure you have everything you need before you start.

- The agent will need to be able to reach your Contrast instance. It can be a local/on-premise instance or a hosted instance. A proxy can be configured if the environment has limited network access.
- It must use [supported versions, frameworks, and tools \(page 120\)](#)
- Understand the [order of precedence \(page 106\)](#)
- You will also need access to a command line interface (with a chosen directory for downloading the agent) and your organization's instance of Contrast

Start the Go agent wizard

1. Select **Add New** at the top of the Contrast web interface.
2. Select the **Application** card.
3. Under Select a language, select **Go**.
4. Select the operating system you are using.

Install the agent

1. Under Select application deployment method, select **Install in a pipeline**.
2. Under Select method to deploy pipelines, select **GitHub** or **Bitbucket**.

3. Under Install an agent:
 - a. Copy the displayed command to add a step to your GitHub action or Bitbucket pipeline.
 - b. Copy the displayed command to the step running your application:

Configure the agent

Under Configure the agent, select the best options to get the configuration details that the agent needs to communicate with Contrast:



NOTE

You have the option of using the [Contrast Agent Configuration Editor \(page 109\)](#) to configure agent settings. Select **Use configuration editor** next to Use connection token or Use API configuration.

1. Select the agent key that you want to use.

If only one key exists, it's preselected.

[Configure agent keys \(page 1162\)](#) describes how to create multiple agent keys.

2. Optionally, select **Register the application**.
When you register an application, you can assign it to groups for access control. You can also assign tags and metadata to the application.
3. Select **Use Connection Token**.
This configuration method uses a single variable to specify authentication values. The Connection Token is the recommended configuration option.
Copy the displayed code and set the variable in a location where the agent has access to it.

Verify the agent deployment

1. Interact with your application as you normally would. For example, interact with your application's web interface or send API commands
2. Go to the Contrast web interface and select **Applications** in the header.
3. Verify that you see the name of your application in the list.
4. Select servers in the header.
5. Verify that you see the name of your local server in the list.

Ansible playbook for Contrast agents

An Ansible playbook lets you use a repeatable process for deploying Contrast agents in your build systems. The Ansible playbook for Contrast automatically installs a Contrast agent in a specific directory under the ownership and permissions of a specified user.

You can use an Ansible playbook for any of the Contrast agents.

Resources

The [Contrast Ansible role](#) provides files that you can use to define the Ansible [role](#) and tasks for Contrast. The instance running this role needs Contrast login credentials and network access to Contrast.

- The `vagrantfile` lets you configure the use of the Ansible playbook to deploy Contrast agents.
- The `defaults/main.yml` file lets you define the Contrast [role variables](#) for the Ansible playbook:

```
contrast_api_key: <apikey>
contrast_service_key: <servicekey>
contrast_username: <email@yourcompany.com>
contrast_teamserver_url: <https://app.contrastsecurity.com>
contrast_teamserver_organization: <organizationname>
contrast_agent_type: java?jvm=1_6
contrast_agent_path_group: vagrant
contrast_agent_path_owner: vagrant
contrast_agent_path: "/opt"
```

- Replace <apikey> with the API key from the Contrast web interface
- Replace <service key> with the service key from the Contrast web interface
- Replace <email@yourcompany.com> with the Contrast username that you received when you activated your account.
- Replace <organizationname> with the name of your Contrast organization.
- The `tasks/main.yml` file lets you define the tasks for the role.

Ansible Playbook example

This example shows how to apply the Contrast role to any server host that uses the playbook.

```
- hosts: servers
  roles:
    - { role: contrast }
```

Install .NET agents with infrastructure as code tools

You can use any of these infrastructure tools for .NET Core and .NET Framework agents.

- [Ansible playbook \(page 94\)](#)
- [Terraform \(page 95\)](#)
- [Azure Resource Manager \(ARM\) \(page 98\)](#)

Install .NET agents with Terraform

Use this procedure to install .NET Framework and .NET Core agents when using Terraform to deploy to Azure. You might need to customize this procedure for your environment.

Site extensions are the best way to deploy the Contrast agent to an Azure app service. You can only do this using the Azure Portal, an ARM policy, or the Azure API. The Terraform method described in this procedure uses the latter two methods directly or indirectly.

Before you begin

- Verify that Contrast supports your preferred OS and runtime stack for the .NET Framework and .NET Core agents running in an Azure App Service:
 - [Supported technologies for .NET Core \(page 271\)](#)
 - [Supported technologies for .NET Framework \(page 212\)](#)
- Ensure you have met these requirements:
 - Login access to Contrast
 - Console access to a system where Terraform and the Azure CLI are installed
 - Login access to Azure Portal, including `az` login from the Azure CLI
 - Python is installed on the system where these commands are run
 - Included the Contrast agent as a part of [Azure App Service \(page 278\)](#)

Step 1: Configure the agent

1. Download a configuration file from Contrast:
 - a. In the Contrast web interface, select **Add New**.
 - b. Select the Application card.
 - c. Follow the displayed instructions to get the required values and download a YAML configuration file.
2. In the YAML configuration file, set the following values:
 - .NET Core agent

```
CORECLR_ENABLE_PROFILING:1
CORECLR_PROFILER:{8B2CE134-0948-48CA-A4B2-80DDAD9F5791}
CORECLR_PROFILER_PATH_32:
  D:\\home\\SiteExtensions\\Contrast.NetCore.Azure.SiteExtension\\
  \\ContrastNetCoreAppService\\contrast\\runtimes\\win-x86\\native\\
  \\ContrastProfiler.dllCORECLR_PROFILER_PATH_64:
  D:\\home\\SiteExtensions\\Contrast.NetCore.Azure.SiteExtension\\
  \\ContrastNetCoreAppService\\contrast\\runtimes\\win-x64\\native\\
  \\ContrastProfiler.dll
```

- .NET Framework agent

```
COR_ENABLE_PROFILING: 1
COR_PROFILER: {EFEB8EE0-6D39-4347-A5FE-4D0C88BC5BC1}
COR_PROFILER_PATH_32: D:\\home\\
  \\SiteExtensions\\Contrast.NET.Azure.SiteExtension\\ContrastAppService\\
  \\ContrastProfiler-32.dllCOR_PROFILER_PATH_64: D:\\home\\
  \\SiteExtensions\\Contrast.NET.Azure.SiteExtension\\ContrastAppService\\
  \\ContrastProfiler-64.dll
```

Step 2: Configure site extensions with Terraform

Because site extension deployment is only natively supported using the Azure portal, Azure ARM policies, and Azure API, Terraform is a convenient command line method to add or remove site extensions. It uses an ARM policy to set up the extension as shown in the examples.

Use this procedure to instrument your application.

1. Verify that the YAML configuration file you prepared in step 1 is named `contrast_security.yaml`.
2. Install Terraform from here: <https://www.terraform.io/downloads.html>.
3. Install PyYAML using this command:

```
pip install PyYAML
```

4. Install the Azure CLI tools from this location: <https://docs.microsoft.com/en-us/cli/azure/install-azure-cli>
5. Log in to Azure to make sure you cache your credentials using `az login`
6. Use this parsing script `parseyaml.py` to pull values out of the Contrast YAML file and add them to the provisioned Azure App Service with this command:

```
import yaml,
  jsonwith open('./contrast_security.yaml') as f:
    config = yaml.load(f)
    print(json.dumps(config['api']))
```

7. Modify the Terraform document called `main.tf` as follows:

```
provider "azurerm" {
  features {}
}
```

```
# Create a resource group
resource "azurerm_resource_group" "personal" {
  name      = <name>
  location  = <location>
}

# Create an app service plan
resource "azurerm_app_service_plan" "app_service-plan" {
  name                = <name>
  resource_group_name = azurerm_resource_group.personal.name
  location             = <location>
}

# Create an app service
resource "azurerm_app_service" "app_service" {
  name                = <name>
  location             = <location>
  resource_group_name = azurerm_resource_group.personal.name
  app_service_plan_id = azurerm_app_service_plan.app_service-plan.id
  site_config {
    dotnet_framework_version = "v4.0"
    default_documents        = ["Default.aspx"]
  }
}

# CONTRAST .NET FRAMEWORK AGENT SETUP
# Contrast env vars will be passed to the app service here.
app_settings = {
  "COR_ENABLE_PROFILING"      = "1"
  "COR_PROFILER"              = "{EFEB8EE0-6D39-4347-A5FE-4D0C88BC5BC1}"
  "COR_PROFILER_PATH_32"      = "D:\\home\\SiteExtensions\\Contrast.NET.Azure.SiteExtension\\ContrastAppService\\ContrastProfiler-32.dll"
  "COR_PROFILER_PATH_64"      = "D:\\home\\SiteExtensions\\Contrast.NET.Azure.SiteExtension\\ContrastAppService\\ContrastProfiler-64.dll"
  "CONTRAST_INSTALL_DIRECTORY" = "D:\\home\\SiteExtensions\\Contrast.NET.Azure.SiteExtension\\ContrastAppService\\"
  "CONTRAST_API_URL"           =
data.external.yaml.result.url
  "CONTRAST_API_USER_NAME"     =
data.external.yaml.result.user_name
  "CONTRAST_API_SERVICE_KEY"   =
data.external.yaml.result.service_key
  "CONTRAST_API_API_KEY"       =
data.external.yaml.result.api_key
  # USE THESE SETTING FOR .NET CORE AGENT
  #"CORECLR_ENABLE_PROFILING" = 1
  #"CORECLR_PROFILER"         = {8B2CE134-0948-48CA-A4B2-80DDAD9F5791}
  #"CORECLR_PROFILER_PATH_32" = D:\\home\\SiteExtensions\\Contrast.NetCore.Azure.SiteExtension\\ContrastNetCoreAppService\\contrast\\runtimes\\win-x86\\native\\ContrastProfiler.dll
  #"CORECLR_PROFILER_PATH_64" = D:\\home\\SiteExtensions\\Contrast.NetCore.Azure.SiteExtension\\ContrastNetCoreAppService\\contrast\\runtimes\\win-x64\\native\\ContrastProfiler.dll
}
}

#Extract the connection from the normal yaml
```

```

file to pass to the app container
data "external" "yaml" {
  program = [var.python_binary, "${path.module}/parseyaml.py"]
}
# Deploy the extension template
resource "azurerm_template_deployment" "extension" {
  name                       = <name>
  resource_group_name       = <resource_group_name>
  template_body              = <<BODY
{
  "$schema": "https://schema.management.azure.com/schemas/
2015-01-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "siteName": {
      "type": "string",
      "metadata": {
        "description": "The Azure App Service Name"
      }
    },
    "extensionName": {
      "type": "string",
      "metadata": {
        "description": "The Site Extension Name."
      }
    }
  },
  "resources": [
    {
      "type": "Microsoft.Web/sites/siteextensions",
      "name": "[concat(parameters('siteName'),
        '/', parameters('extensionName'))]",
      "apiVersion": "2019-08-01",
      "location": "[resourceGroup().location]"
    }
  ]
}
BODY parameters = {
  "siteName"           = azurerm_app_service.<app_service>.name
  #.NET Framework
  "extensionName"      = "Contrast.NET.Azure.SiteExtension"
  #.NET Core
  # "extensionName"     = "Contrast.NetCore.Azure.SiteExtension"
}
  deployment_mode      = "Incremental"
}

```

Install .NET agents with Azure Resource Manager

This topic describes the most popular methods for instrumenting .NET Framework and .NET Core applications through automation with Azure Resource Manager (ARM) templates and Azure.

You can only configure site extensions either directly through the Azure Portal with an ARM policy or through Azure's REST API. All methods described in this topic use one of these methods. Only REST API and Azure ARM policies allow for automated deployments.

Before you begin

- Verify that Contrast supports your preferred OS and runtime stack for the .NET Framework and .NET Core agents
 - [Supported technologies for .NET Core \(page 271\)](#)
 - [Supported technologies for .NET Framework \(page 212\)](#)
- Ensure you have login access to Contrast.
- Only configure a Contrast site extension with backend HTTP services and not WebJobs or UI app services.
- The methods in this topic do not work if you are deploying an App service using Docker.

Step 1: Download an agent configuration file

1. Download a configuration file from Contrast.
 - a. In the Contrast web interface, select **Add New**.
 - b. Select the Application card.
 - c. Follow the displayed instructions to get the required values and download a YAML configuration file.
2. To automate your ARM templates more fully, get your Contrast API credentials, either from the `contrast_security.yml` file that you downloaded or from the Contrast web interface (**user menu > Organization Settings > Agent**). These credentials are:

Configuration setting	Contrast label
CONTRAST__API__API_KEY	API Key
CONTRAST__API__URL	Contrast Agent URL
CONTRAST__API__USER_NAME	Contrast Agent Username
CONTRAST__API__SERVICE_KEY	Agent Service Key

Step 2: Edit the ARM template

Add the highlighted Contrast configuration values to the ARM template, as shown in the following examples.

[.NET Core-specific configuration \(page 290\)](#) and [.NET Framework-specific configuration \(page 228\)](#) provide additional configuration details.

- .NET Core

```
{
  "$schema": "https://schema.management.azure.com/schemas/2015-01-01/
deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "sites_name": {
      "defaultValue": "APP_NAME",
      "type": "String"
    }
  },
  "variables": {
  },
  "resources": [
    {
      "type": "Microsoft.Web/sites",
      "apiVersion": "2018-11-01",
      "name": "[parameters('sites_name')]",
      "location": "East US",
```

```

"kind": "app",
"properties": {
  "siteConfig": {
    "appSettings": [
      {
        "name": "CONTRAST_API_API_KEY",
        "value": "<CONTRAST_API_API_KEY>",
        "slotSetting": false
      },
      {
        "name": "CONTRAST_API_SERVICE_KEY",
        "value": "<CONTRAST_API_SERVICE_KEY>",
        "slotSetting": false
      },
      {
        "name": "CONTRAST_API_URL",
        "value": "<CONTRAST_API_URL>",
        "slotSetting": false
      },
      {
        "name": "CONTRAST_API_USER_NAME",
        "value": "<CONTRAST_API_USER_NAME>",
        "slotSetting": false
      },
      {
        "name": "CORECLR_ENABLE_PROFILING",
        "value": "1",
        "slotSetting": false
      },
      {
        "name": "CORECLR_PROFILER",
        "value": "{8B2CE134-0948-48CA-
A4B2-80DDAD9F5791}",
        "slotSetting": false
      },
      {
        "name": "CORECLR_PROFILER_PATH_32",
        "value": "D:\\home\\SiteExtensions\\
\\Contrast.NetCore.Azure.SiteExtension\\ContrastNetCoreAppService\\
\\contrast\\runtimes\\win-x86\\native\\ContrastProfiler.dll",
        "slotSetting": false
      },
      {
        "name": "CORECLR_PROFILER_PATH_64",
        "value": "D:\\home\\SiteExtensions\\
\\Contrast.NetCore.Azure.SiteExtension\\ContrastNetCoreAppService\\
\\contrast\\runtimes\\win-x64\\native\\ContrastProfiler.dll",
        "slotSetting": false
      }
    ]
  }
},
"resources": [
  {
    "name": "Contrast.NetCore.Azure.SiteExtension",

```



```

        "type": "siteextensions",
        "apiVersion": "2018-02-01",
        "dependsOn": [
            "[resourceId('Microsoft.Web/Sites',
parameters('sites_name'))]"
        ]
    }
]
}
]
}

```

- .NET Framework

```

{
  "$schema": "https://schema.management.azure.com/schemas/2015-01-01/
deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "sites_name": {
      "defaultValue": "APP_NAME",
      "type": "String"
    }
  },
  "variables": {
  },
  "resources": [
    {
      "type": "Microsoft.Web/sites",
      "apiVersion": "2018-11-01",
      "name": "[parameters('sites_name')]",
      "location": "East US",
      "kind": "app",
      "properties": {
        "siteConfig": {
          "appSettings": [
            {
              "name": "CONTRAST_API_API_KEY",
              "value": "<CONTRAST_API_API_KEY>",
              "slotSetting": false
            },
            {
              "name": "CONTRAST_API_SERVICE_KEY",
              "value": "<CONTRAST_API_SERVICE_KEY>",
              "slotSetting": false
            },
            {
              "name": "CONTRAST_API_URL",
              "value": "<CONTRAST_API_URL>",
              "slotSetting": false
            },
            {
              "name": "CONTRAST_API_USER_NAME",
              "value": "<CONTRAST_API_USER_NAME>",
              "slotSetting": false
            }
          ]
        }
      }
    }
  ]
}

```

```

        "name": "COR_ENABLE_PROFILING",
        "value": "1",
        "slotSetting": false
      },
      {
        "name": "COR_PROFILER",
        "value": "{EFE88EE0-6D39-4347-A5FE-4D0C88BC5BC1}",
        "slotSetting": false
      },
      {
        "name": "COR_PROFILER_PATH_32",
        "value": "D:\\home\\SiteExtensions\\Contrast.NET.Azure.SiteExtension\\ContrastAppService\\ContrastProfiler-32.dll",
        "slotSetting": false
      },
      {
        "name": "COR_PROFILER_PATH_64",
        "value": "D:\\home\\SiteExtensions\\Contrast.NET.Azure.SiteExtension\\ContrastAppService\\ContrastProfiler-64.dll",
        "slotSetting": false
      }
    ]
  },
  "resources": [
    {
      "name": "Contrast.NET.Azure.SiteExtension",
      "type": "siteextensions",
      "apiVersion": "2018-02-01",
      "dependsOn": [
        "[resourceId('Microsoft.Web/Sites', parameters('sites_name'))]"
      ]
    }
  ]
}

```

Step 3: Deploy the application from the ARM template

Use one of these methods in the Azure documentation:

- Command line or CLI: [Use ARM deployment templates with Azure CLI](#).
- Azure Portal: [Edit and deploy ARM templates](#).

Configure an agent

When you install an agent, you must configure it so that it recognizes your application and can communicate information back to Contrast.

Configuration follows this [order of precedence \(page 106\)](#).

**NOTE**

An expired license or exceeding a license quota disables all agent behavior regardless of configuration.

Steps

1. **Recommended:** Use the Agent token to configure the required authentication variables (you can find this value in Contrast) (page 104).

```
api:
  token: <token-value>
```

Where the <token-value> is a base64 encoded JSON object containing the url, api_key, service_key, and user_name values, allowing you to set them in a single variable.

Legacy settings: If you are using an older version of the agent, configure these authentication variables:

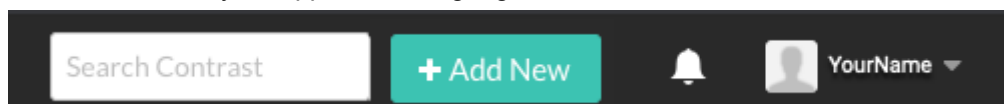
```
api:
  url: https://<environment>-agents.contrastsecurity.com
  user_name: contrast_user
  api_key: demo
  service_key: demo
```

where:

- **url:** Address of the Contrast installation you would like your agent to report to. Defaults to: `https://<environment>-agents.contrastsecurity.com/Contrast`
- **user_name:** Contrast user account
- **api_key:** Your organization's API key
- **service_key:** Contrast user account service key

You can set these authentication variables with either:

1. Environment variables.
The simplest way to configure these values is to use an agent wizard (In the Contrast web interface, select **Add New**, select the **Application** card, and follow the instructions for your language). You can open the [Contrast agent configuration editor \(page 109\)](#) from the agent wizard to configure these values.
2. YAML configuration file.
 - You can download a [YAML configuration file \(page 107\)](#) that is pre-populated with your organization keys. Select **Add new** in the Contrast web interface, select the **Application** card, and choose your application language to find a download link.



- You can also configure the file with the [Contrast agent configuration editor \(page 109\)](#) which you can open from the agent wizard.
3. Other methods native to the language and tools you are using, such as system properties or command line flags. Refer to the individual documentation pages for more details.

**NOTE**

See [Contrast agent configuration editor](#) to view a full list of options and their default values.

2. Configure any additional variables.

- Use [session metadata \(page 616\)](#) to filter vulnerabilities and route information for a specific branch, build, committer, or repository.
- Use [application metadata \(page 1174\)](#) to filter applications by custom values.

When you add the necessary configuration settings to your agent configuration file, the agent reports this information along with the rest of your standard vulnerability data to Contrast. Look [here \(page 111\)](#) for the full list of configuration values and what they do beyond the necessary values described above.

Find the agent keys

Agent keys are values that authenticate the agents. Unlike user and service level authentication, they can only be used to report data collected for your organization; they cannot be used for UI access or data retrieval. Your organization can have multiple agent keys with different authentication credentials.

By using different agent keys for each team or service instead of a single, shared agent key, you can reduce security risks and gain more granular control over agent deployments.

**IMPORTANT**

If you download a YAML configuration file from Contrast instead of using the configuration that an agent wizard creates (select **Add new** in the top right, select **Live Application**, and go to the **Configuration** step), the file is pre-populated with the agent keys for a specified agent key name.

If you create your own YAML file, you'll need to add the keys yourself.

Contrast provides a default agent key. [Configure agent keys \(page 1162\)](#) describes how to add multiple agent keys.

Preferred keys

The preferred key to use when installing agents is the Agent token. The Agent token replaces these legacy keys: Agent service key, Agent username, API key, and Contrast URL.

The agent versions that support the Agent token are:

- Java 6.10.1 or later
- .NET Framework 51.0.40 or later
- .NET Core 4.2.22 or later
- Node.js 5.15.0 or later
- Python 8.6.0 or later
- PHP 1.34.0 or later
- Go 6.11.0 or later

Legacy keys

For older agents, these keys are required when installing agents:

- Agent key name (API__user_name)
- Agent service key
- API key

This API key is for all agents. For the API key to use with custom scripts, use the API key under User settings.

- Contrast URL

Steps

1. Select **User name > Organization settings** in the top right corner.
2. Select **Agent keys**.



NOTE

If you don't see the keys on this page, it may mean that a license has not been applied to your organization. [Contact Support](#) for help with this.

3. Select a key name.

Organization Settings

Organization
Groups
Users
Access Control
Audit Log
Security
Agent Keys
Single Sign-On
Integrations
Servers
Applications
Notifications
Score Settings
Scan projects

Agent Keys

Use agent keys when configuring agents to communicate the Contrast. Learn how in our [API documentation](#).

Search key name + Add key name

KEY NAME	CREATED	LAST SEEN	ACTIONS
Michael-dev-team-1	1/17/2025, 2:42:04 PM		
Michaels-dev-team-beta	1/17/2025, 3:40:35 PM		

Page 1 of 1 Showing 1-3 out of 3 results Results per page 10 20

4. Copy the keys you need.

Agent Keys

Agent Keys

Agent Key Name (API username)

agent_6dev-team-1

Agent Token

Legacy Agent Keys ▾

Looking for your API keys? Go to [User Settings](#).

- For newer agents, copy the agent key name and the agent token.
- For older agent, select **Legacy agent keys** and copy the displayed keys.
 - The Contrast URL is `https://<environment>-agents.contrastsecurity.com/` Contrast, or the URL of your on-premises or private cloud instance.
 - You can **Rotate** agent keys to generate new keys if your credentials have been compromised.

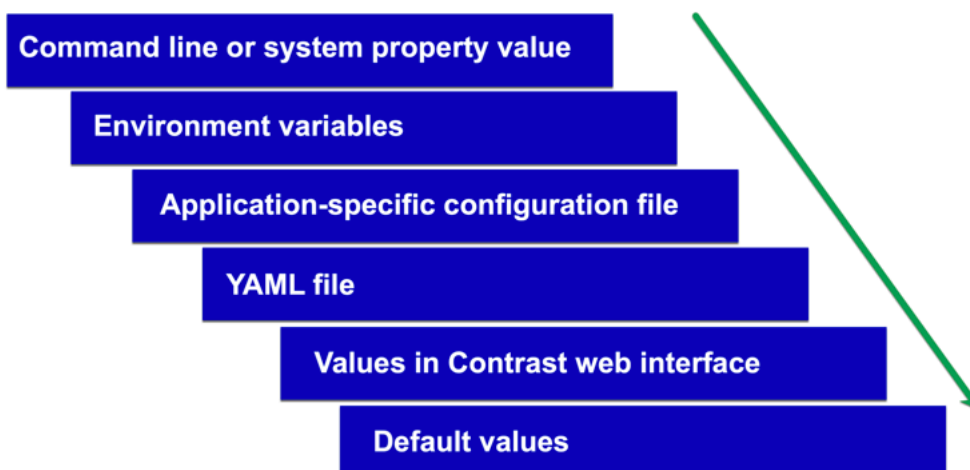


IMPORTANT

Rotating agent service keys will take all agents offline. Your applications will still function, but data will not be sent to Contrast. To begin using the new credentials, reconfigure the agents and restart your applications. You can use a credential management system to coordinate this change among your systems.

Order of precedence

Active configuration values are determined using the following order of precedence:



1. An expired license or exceeding a license quota disables all agent behavior regardless of configuration.
2. Command line or system property value (if appropriate for the language you are using).
For example: `-Dcontrast.enable`
3. [Environment variables \(page 110\)](#).
For example: `CONTRAST__ENABLE`
4. An application-specific configuration file (.NET Framework only).
For example: [web.config \(page 229\)](#)
5. Configuration values in a [YAML file \(page 107\)](#) are pulled from all files, taking the value from the highest precedence file.
For instance, if `contrast_security.yaml` in the current working directory has the application-specific value for `application.tags` and the file in `/etc/contrast/contrast_security.yaml` has the organization-level connection details, the agent would have access to both. If `/etc/contrast/contrast_security.yaml` also had an `application.tags` default value, only the value in the current working directory's configuration, as a higher precedence, would be read; the two values of `application.tags` are not combined.
 - a. A YAML file indicated by the user
For example:
 - Java: the `contrast.config.path` [system property \(page 150\)](#)
 - Any agent: the `CONTRAST_CONFIG_PATH` environment variable.
 - b. A `contrast_security.yaml` file in the current working directory (all agents except Java)
For example: `./contrast_security.yaml`
 - c. A `contrast_security.yaml` file in the application's configuration directory (Ruby and Python only)
For example:
 - Ruby on Rails: `./config/contrast_security.yaml`
 - Django: `./settings/contrast_security.yaml`
 - d. A `contrast_security.yaml` file in an agent-specific configuration directory. For agents that use a service, use this directory if you need to use separate YAML files for agent and service.
For example:
 - `/etc/contrast/agentname/contrast_security.yaml` (where *agentname* is one of: *dotnet*, *go*, *java*, *node*, *python*, *ruby*, or *webserver*)
 - `%ProgramData%\contrast\agentname\contrast_security.yaml` (where *agentname* is one of: *dotnet*, *dotnet-core*, *java*, *node*, *python*, *ruby*, or *webserver*)
 - e. A `contrast_security.yaml` file within the server's `/etc/contrast` directory (all agents except .NET Framework, and .NET Core). For agents that use a service, use this directory if you need to share YAML files between agent and service.
For example:
 - `/etc/contrast/contrast_security.yaml`
 - `%ProgramData%\contrast\contrast_security.yaml`
6. Values set in the Contrast web interface.
For example: Server mode toggles for Assess and Protect, which map to `assess.enable` and `protect.enable`
7. The default value set by Contrast Security.

See also

[Additional configuration \(page 111\)](#)

YAML configuration

You can use a YAML configuration file to set configuration properties for your agent. These values can be overridden with environment variables or command line arguments.

While all Contrast agents share the same property formatting in YAML configuration files, each agent must use its own specified file as there are unique properties that apply to each agent.

The configuration file must be called *contrast_security.yaml* and placed properly in the [load path \(page 106\)](#).

When you download the agent configuration file from Contrast, it will contain all the basic properties required for your instance of Contrast. If you create your configuration file, you must add [these keys \(page 104\)](#) yourself.

The minimum required *contrast_security.yaml* content for the latest version of the agents should look like this:

```
api:
  token: <token-value>
```

Where the <token-value> is a base64 encoded JSON object containing the url, api_key, service_key, and user_name values, allowing you to set them in a single variable.

Legacy settings: For older agents, the minimum required *contrast_security.yaml* content for all agents should look like this:

```
api:
  url: https://<environment>-agents.contrastsecurity.com/Contrast
  user_name: contrast_user
  api_key: demo
  service_key: demo
```



TIP

- Use the [Contrast agent configuration editor \(page 109\)](#) to create or upload a YAML configuration file, validate YAML and get setting recommendations.
- Since YAML is a natural superset of JSON, you can also configure your agent using JSON in your YAML file.

You can use these YAML templates to create a *contrast_security.yaml* for each agent:

- [Java \(page 150\)](#)
- [.NET Framework \(page 230\)](#)
- [.NET Core \(page 292\)](#)
- [Node.js \(page 353\)](#)
- [PHP \(page 397\)](#)
- [Python \(page 413\)](#)
- [Ruby \(page 474\)](#)
- [Go \(page 535\)](#)



CAUTION

Take care when editing the YAML template as it relies on whitespace, and uses spaces but not tabs. Configuration guidance is provided in the template as comments. (A space followed by the pound sign "#" starts a comment.)

Use the Contrast agent configuration editor

The [Contrast agent configuration editor](#) is a web application that can be used to validate and generate the configuration for Contrast agents.

Before you begin

- Use this editor to help edit, validate, and generate configurations for Contrast agents.
- If you already have a YAML file, you can open it in the editor by selecting **Import**.
- The editor executes entirely in the browser and any sensitive information such as your API key won't leave the local machine.

Steps

1. Open the [Contrast YAML configuration editor](#) in your browser.

The screenshot shows the Contrast agent configuration editor interface. At the top, there's a header with the Contrast logo, a dropdown menu set to ".NET Core Agent", and buttons for "Validate using the", "Reset", "Import", "Export", and "Share". Below the header is a large text area containing a YAML configuration template. The template includes fields for API settings (url, api_key, service_key, user_name), inventory tags, application name, and server tags. To the right of the text area is a "Filter Options" panel with a checkbox for "Show advanced settings" and a list of configuration fields with their descriptions. At the bottom, there's a table with two columns: "Type" and "Message", and a "Line number" column. The table contains two rows of messages indicating placeholder values that should be replaced.

```
1  api:
2    url: https://app.contrastsecurity.com
3    api_key: TODO
4    service_key: TODO
5    user_name: TODO
6  inventory:
7    tags: TODO
8  application:
9    name: TODO
10   tags: TODO
11  server:
12   tags: TODO
13
```

Filter Options

☐ Show advanced settings

api.url
Set the URL for the Contrast UI.

api.api_key
Set the API key needed to communicate with the Contrast UI.

api.service_key
Set the service key needed to communicate with the Contrast UI. It is used to calculate the Authorization header.

api.user_name
Set the user name used to communicate with the Contrast UI. It is used to calculate the Authorization header.

inventory.tags
Apply a list of labels to libraries. Labels must be formatted as a comma-delimited list. Example - label1, label2, label3

assess.tags
Apply a list of labels to vulnerabilities and preflight messages. Labels must be formatted as a comma-delimited list. Example - label1, label2, label3

application.name
Override the reported application name. Note - On Java systems where multiple, distinct applications may be served by a single process, this configuration causes the agent to report all discovered applications as one application with the given name.

application.group
Add the name of the application group with which this application

Type	Message	Line number
TODO	Placeholder value 'TODO' should be replaced for setting: api.api_key	3
TODO	Placeholder value 'TODO' should be replaced for setting: api.service_key	4

2. Either import an existing YAML file by selecting **Import**, or paste your YAML content in the main window.
3. As you edit text, an error warning will appear if you enter invalid YAML. Select an option from the **Validate using the** list for agent-specific YAML validation. These types of validation are performed:
 - **YAML syntax validation** verifies that the text can be parsed as YAML. Invalid YAML will result in an **error** that prevents further validation.
 - **Setting key validation** verifies that the YAML nodes represent setting keys supported by the selected agent. An unrecognized setting key will result in a **warning**.
 - **Setting value validation** verifies that the YAML values match type expectations including boolean, numeric, and enum (e.g., log level). Invalid values will result in a **warning**.
 - **Setting compatibility validation** verifies that specific incompatible settings are not both present. This is currently limited to `application.session_id` and `application.session_metadata` settings. Incompatible settings will result in a **warning**.
 - **Placeholder value validation** notifies the user when a setting has the placeholder value `TODO`. Placeholder value will result in a **note**.Select the **error**, **warning**, or **note** in the list of issues to move the text editor's cursor to the start of text causing the issue.
4. Use the panel on the right to search for available settings with descriptions. Select the plus sign (+) to add that setting to your YAML file. Adding new settings using this feature will format the YAML which may re-order nodes and will remove any extra whitespace in the YAML. Click **Reset** to go back to the original file settings.

**NOTE**

YAML generation is disabled when the YAML in the text editor has a syntax error or is not valid YAML.

5. When you are finished, export your configuration file either as a YAML or environment variables. You can also share the file with other collaborators.

**NOTE**

At this time, the Contrast configuration editor executes completely offline (meaning after first visit, the page is accessible without an internet connection). Updates are downloaded in the background automatically. Upgrading to newer versions requires closing all agent configuration app tabs; refreshing is not enough to activate the new version.

Environment variables

You can configure the agent with any of the supported properties through environment variables.

The environment variables need to be set before the agent starts up, and in a location where the agent has access to it. Environment variables can be set within the same process or system-wide.

**IMPORTANT**

If you set system-wide environment variables, this may impact other Contrast agents running on the same server.

You can convert any Contrast property between command line, YAML and an environment variable.

To convert a command line formatted variable to an environment variable, replace the path segment delimiters (.) with double underscores (__).

To convert a YAML formatted variable to an environment variable, start with the top-level property and separate every nested property with a double underscore (__).

Then, prepend the "contrast" namespace (either `contrast.` or `CONTRAST__`).

Environment variables should be in all caps and have no spaces.

For example:

Command line	YAML property	Environment variable
<code>contrast.server.name</code>	<code>server: name:</code>	<code>CONTRAST__SERVER__NAME</code>
<code>contrast.api.api_key</code>	<code>api: api_key:</code>	<code>CONTRAST__API__API_KEY</code>

You can see a list of all supported properties for each agent in their respective YAML templates:

- [Java \(page 150\)](#)
- [.NET Framework \(page 230\)](#)
- [.NET Core \(page 292\)](#)
- [Node.js \(page 353\)](#)
- [PHP \(page 397\)](#)
- [Python \(page 413\)](#)
- [Ruby \(page 474\)](#)
- [Go \(page 535\)](#)

See also

[Additional configuration \(page 111\)](#) for more details about the environment variables used for configuration.

Additional configuration

You can set these common variables with either system properties, environment variables, a YAML file, or default values.

Additional configuration values set with environment variables

Use these common variables to configure your system.

See the [Contrast YAML Configuration Editor](#) for a complete list as this can be updated with language-specific and advanced settings.

Environment variable	Description	Language
CONTRAST__API__TOKEN	Set these values needed to communicate with Contrast: URL, API key, service key, and user name. This variable is the preferred method for setting the authentication credentials.	Latest versions of Java, .NET Framework, .NET Core, Node.js, Python, Ruby, PHP, and Go
CONTRAST__API__URL	Set the URL for Contrast.	Java, .NET Framework, .NET Core, Node.js, Python, PHP, and Go
CONTRAST__API__API_KEY	Set the API key needed to communicate with Contrast.	Java, .NET Framework, .NET Core, Node.js, Python, Ruby, PHP, and Go
CONTRAST__API__SERVICE_KEY	Set the service key needed to communicate with Contrast. It is used to calculate the Authorization header.	Java, .NET Framework, .NET Core, Node.js, Python, Ruby, PHP, and Go
CONTRAST__API__USER_NAME	Set the user name used to communicate with Contrast. It is used to calculate the Authorization header.	Java, .NET Framework, .NET Core, Node.js, Python, Ruby, PHP, and Go
CONTRAST__INVENTORY__TAGS	Apply a list of labels to libraries. Labels must be formatted as a comma-delimited list. Example: label1, label2, label3.	Java, .NET Framework, .NET Core, Node.js, Python, Ruby, and PHP
CONTRAST__ASSESS__TAGS	Apply a list of labels to vulnerabilities and preflight messages. Labels must be formatted as a comma-delimited list. Example: label1, label2, label3.	Java, .NET Framework, .NET Core, Node.js, Python, Ruby, PHP, and Go
CONTRAST__APPLICATION__NAME	Override the reported application name. Note: On Java systems where multiple, distinct applications may be served by a single process, this configuration causes the agent to report all discovered applications as one application with the given name.	Java, .NET Framework, .NET Core, Node.js, Python, Ruby, PHP, and Go
CONTRAST__APPLICATION__GROUP	Add the name of the application group with which this application should be associated in Contrast.	Java, .NET Framework, .NET Core, Node.js, Python, Ruby, PHP, and Go
CONTRAST__APPLICATION__CODE	Add the application code this application should use in Contrast.	Java, .NET Framework, .NET Core, Node.js, Python, Ruby, PHP, and Go
CONTRAST__APPLICATION__VERSION	Override the reported application version.	Java, .NET Framework, .NET Core, Node.js, Python, Ruby, PHP, and Go
CONTRAST__APPLICATION__TAGS	Apply labels to an application. Labels must be formatted as a comma-delimited list. Example: label1, label2, label3.	Java, .NET Framework, .NET Core, Node.js, Python, Ruby, PHP, and Go
CONTRAST__SERVER__NAME	Override the reported server name.	Java, .NET Framework, .NET Core, Node.js, Python, Ruby, PHP, and Go
CONTRAST__SERVER__ENVIRONMENT	Override the reported server environment. Valid values include QA, PRODUCTION and DEVELOPMENT.	Java, .NET Framework, .NET Core, Node.js, Python, Ruby, PHP, and Go
CONTRAST__SERVER__TAGS	Apply a list of labels to the server. Labels must be formatted as a comma-delimited list. Example: label1, label2, label3.	Java, .NET Framework, .NET Core, Node.js, Python, Ruby, PHP, and Go

Additional configuration values set by web.config

If you use .NET Framework or .NET Core with any of the following, you will also need to configure these variables.

Platform	Variable set
Web.config (.NET Core IIS Module)	<pre> <environmentVariable name="CONTRAST__API__URL" value="https:// app.contrastsecurity.com/Contrast/ " /> <environmentVariable name="CONTRAST__API__API_KEY" value="" /> <environmentVariable name="CONTRAST__API__SERVICE_KEY" value="" /> <environmentVariable name="CONTRAST__API__USER_NAME" value="" /> <environmentVariable name="CONTRAST__INVENTORY__TAGS" value="" /> <environmentVariable name="CONTRAST__ASSESS__TAGS" value="" /> <environmentVariable name="CONTRAST__APPLICATION__NAME" value="" /> <environmentVariable name="CONTRAST__APPLICATION__GROUP" value="" /> <environmentVariable name="CONTRAST__APPLICATION__CODE" value="" /> <environmentVariable name="CONTRAST__APPLICATION__VERSION" value="" /> <environmentVariable name="CONTRAST__APPLICATION__TAGS" value="" /> <environmentVariable name="CONTRAST__APPLICATION__METADATA" value="" /> <environmentVariable name="CONTRAST__APPLICATION__SESSION_ID" value="" /> <environmentVariable name="CONTRAST__APPLICATION__SESSION_METADATA" value="" /> <environmentVariable name="CONTRAST__SERVER__NAME" value="localhost" /> <environmentVariable name="CONTRAST__SERVER__ENVIRONMENT" value="development" /> <environmentVariable name="CONTRAST__SERVER__TAGS" value="" /> </pre>

Platform	Variable set
Azure App Service	<pre>[{ "name": "CONTRAST__API__URL", "value": "https://app.contrastsecurity.com/Contrast/ " }, { "name": "CONTRAST__API__API_KEY", "value": "" }, { "name": "CONTRAST__API__SERVICE_KEY", "value": "" }, { "name": "CONTRAST__API__USER_NAME", "value": "" }, { "name": "CONTRAST__INVENTORY__TAGS", "value": "" }, { "name": "CONTRAST__ASSESS__TAGS", "value": "" }, { "name": "CONTRAST__APPLICATION__NAME", "value": "" }, { "name": "CONTRAST__APPLICATION__GROUP", "value": "" }, { "name": "CONTRAST__APPLICATION__CODE", "value": "" }, { "name": "CONTRAST__APPLICATION__VERSION", "value": "" }, { "name": "CONTRAST__APPLICATION__TAGS", "value": "" }, { "name": "CONTRAST__APPLICATION__METADATA", "value": "" }, { "name": "CONTRAST__APPLICATION__SESSION_ID", "value": "" }, { "name": "CONTRAST__APPLICATION__SESSION_METADATA", "value": "" }, { "name": "CONTRAST__SERVER__NAME", "value": "localhost" }, { "name": "CONTRAST__SERVER__ENVIRONMENT", "value": "development" }, { "name": "CONTRAST__SERVER__TAGS", "value": "" }]</pre>

Additional configuration values set by system properties

Configuration value	Languages
-Dcontrast.api.url	Java, .NET Framework, .NET Core, Node.js, PHP, Python, Ruby, and Go
-Dcontrast.api.api_key	Java, .NET Framework, .NET Core, Node.js, PHP, Python, Ruby, and Go
-Dcontrast.api.service_key	Java, .NET Framework, .NET Core, Node.js, PHP, Python, Ruby, and Go
-Dcontrast.api.user_name	Java, .NET Framework, .NET Core, Node.js, PHP, Python, Ruby, and Go
-Dcontrast.inventory.tags	Java, .NET Framework, .NET Core, Node.js, PHP, Python, and Ruby
-Dcontrast.assess.tags	Java, .NET Framework, .NET Core, Node.js, PHP, Python, Ruby, and Go
-Dcontrast.application.name	Java, .NET Framework, .NET Core, Node.js, PHP, Python, Ruby, and Go
-Dcontrast.application.group	Java, .NET Framework, .NET Core, Node.js, PHP, Python, Ruby, and Go
-Dcontrast.application.code	Java, .NET Framework, .NET Core, Node.js, PHP, Python, Ruby, and Go
-Dcontrast.application.version	Java, .NET Framework, .NET Core, Node.js, PHP, Python, Ruby, and Go
-Dcontrast.application.tags	Java, .NET Framework, .NET Core, Node.js, PHP, Python, Ruby, and Go
-Dcontrast.server.name	Java, .NET Framework, .NET Core, Node.js, PHP, Python, Ruby, and Go
-Dcontrast.server.environment	Java, .NET Framework, .NET Core, Node.js, PHP, Python, Ruby, and Go
-Dcontrast.server.tags	Java, .NET Framework, .NET Core, Node.js, PHP, Python, Ruby, and Go

Additional configuration values set in the YAML

Use the YAML to set these additional configuration values.

Property	Description	Languages
contrast.api.url	Set the URL for Contrast.	Java, .NET Framework, .NET Core, Node.js, Python, Ruby, PHP, and Go
contrast.api.api_key	Set the API key needed to communicate with Contrast.	Java, .NET Framework, .NET Core, Node.js, Python, Ruby, PHP, and Go
contrast.api.service_key	Set the service key needed to communicate with Contrast. It is used to calculate the Authorization header.	Java, .NET Framework, .NET Core, Node.js, Python, Ruby, PHP, and Go
contrast.api.user_name	Set the user name used to communicate with Contrast. It is used to calculate the Authorization header.	Java, .NET Framework, .NET Core, Node.js, Python, Ruby, PHP, and Go
contrast.inventory.tags	Apply a list of labels to libraries. Labels must be formatted as a comma-delimited list. Example: label1, label2, label3.	Java, .NET Framework, .NET Core, Node.js, Python, Ruby, and PHP
contrast.assess.tags	Apply a list of labels to vulnerabilities and preflight messages. Labels must be formatted as a comma-delimited list. Example: label1, label2, label3.	Java, .NET Framework, .NET Core, Node.js, Python, Ruby, PHP, and Go
contrast.application.name	Override the reported application name. Note: On Java systems where multiple, distinct applications may be served by a single process, this configuration causes the agent to report all discovered applications as one application with the given name.	Java, .NET Framework, .NET Core, Node.js, Python, Ruby, PHP, and Go

Property	Description	Languages
contrast.application.group	Add the name of the application group with which this application should be associated in Contrast.	Java, .NET Framework, .NET Core, Node.js, Python, Ruby, PHP, and Go
contrast.application.code	Add the application code this application should use in Contrast.	Java, .NET Framework, .NET Core, Node.js, Python, Ruby, PHP, and Go
contrast.application.metadata	Define a set of <code>key=value</code> pairs (which conforms to RFC 2253) for specifying user-defined metadata associated with the application. The set must be formatted as a comma-delimited list of <code>key=value</code> pairs. Example: <code>business-unit=accounting,office=Baltimore</code>	Java, .NET Framework, .NET Core, Node.js, Python, Ruby, PHP, and Go
contrast.application.session_id	Provide the ID of a session that already exists in Contrast. Vulnerabilities discovered by the agent are associated with this session. If an invalid ID is supplied, the agent will be disabled. This option and <code>application.session_metadata</code> are mutually exclusive; if both are set, the agent will be disabled.	Java, .NET Framework, .NET Core, Node.js, Python, Ruby, PHP, and Go
contrast.application.session_metadata	Provide metadata that is used to create a new session ID in Contrast. Vulnerabilities discovered by the agent are associated with this new session. This value should be formatted as <code>key=value</code> pairs (conforming to RFC 2253).	Java, .NET Framework, .NET Core, Node.js, Python, Ruby, PHP, and Go
contrast.application.version	Override the reported application version.	Java, .NET Framework, .NET Core, Node.js, Python, Ruby, PHP, and Go
contrast.application.tags	Apply labels to an application. Labels must be formatted as a comma-delimited list. Example: <code>label1, label2, label3</code> .	Java, .NET Framework, .NET Core, Node.js, Python, Ruby, PHP, and Go
contrast.server.name	Override the reported server name.	Java, .NET Framework, .NET Core, Node.js, Python, Ruby, PHP, and Go
contrast.server.environment	Override the reported server environment. Valid values include QA, PRODUCTION and DEVELOPMENT.	Java, .NET Framework, .NET Core, Node.js, Python, Ruby, PHP, and Go
contrast.server.tags	Apply a list of labels to the server. Labels must be formatted as a comma-delimited list. Example: <code>label1, label2, label3</code> .	Java, .NET Framework, .NET Core, Node.js, Python, Ruby, PHP, and Go

Tags and data

Tags

You may want or need to filter either Applications or Servers based on user-defined criteria. In this case, tags may be desired instead of or in addition to metadata. Tags can be applied to either an [application \(page 612\)](#), [server \(page 912\)](#), [libraries \(page 929\)](#), and/or [vulnerabilities \(page 1028\)](#). These tags can help better organize items and improve search in Contrast.

Metadata

Application metadata can be created during agent configuration to help collect data from applications. You can set up fields to identify specific application owners, business units, locations, or other important pieces of information associated with an application.

Session metadata

You may want or need to filter vulnerability data for specific fields of information. When supplied in the agent configuration, the session metadata can be used as a filter in the [Application Vulnerability \(page 1024\)](#) details (not the Vulnerability Tab).



NOTE

Most Contrast agents now automatically create a unique fingerprint that is equal to that unique build of an agent and populates session metadata values. The key used for this is called `artifactHash`. [Session metadata \(page 616\)](#) lists the supported agent versions for this feature.

Commonly-used fields include:

Name	Value
Commit Hash	<code>commitHash</code>
Committer	<code>committer</code>
Branch Name	<code>branchName</code>
Git Tag	<code>gitTag</code>
Repository	<code>repository</code>
Test Run	<code>testRun</code>
Version	<code>version</code>
Build Number	<code>buildNumber</code>

For more information see [session metadata \(page 616\)](#).

Methods for exercising applications

After you install and configure a Contrast agent, thoroughly exercising your applications ensures that Contrast can provide the most accurate information about vulnerabilities.

These application deployment methods can help to exercise as many routes in your application as possible, depending on your tools and environment.

For best results:	Test requirements
Deploy to a server that receives requests from integration and smoke tests in a CI/CD pipeline. (page 117)	Existing automation tests
Deploy to a server receiving requests from manual testing (page 118)	Users who do manual testing
Deploy to server that receives requests from web application test automation tools (page 118)	Automated tests
Deploy to a server that receives requests from API testing tools (page 118)	API testing tools like Postman
Deploy to server with DAST testing tools (page 118)	DAST tools like Rapid7
Run an open-source crawler (page 119)	Free tools like Zap
Deploy to server that you use in a manual penetration testing environment (page 119)	Either first or third party pen testers exercising applications
Deploy to server that you use for BurpSuite-based penetration tests. (page 119)	Burp (for BurpTrast integration)
Use curl commands with Assess data. (page 120)	Any user who can authenticate an API

Deployment to CI/CD pipeline

This method of exercising applications involves deployment to a server that receives requests from integration and smoke tests in a CI/CD pipeline.

Details

Requirements	Users	Best for...
<ul style="list-style-type: none">Automation tests (regression, integration, smoke, performance tests),A CI tool to manage and orchestrate the automation,A server for CICD.	<p>Users responsible for writing tests, maintaining CI, and instrumenting the CI server.</p> <p>These users are usually a mix of QA, development, and DevOps staff.</p>	All environments

Deployment with manual testing

This method of exercising applications involves deployment to a server that receives requests from manual testing.

Details

Requirements	Users	Best for...
A server you can use for manual testing.	QA testers who run manual tests	Environments that have manual resources available instead of automated testing environments.

Deployment with web application test tools

This method of exercising applications involves deployment to a server that receives requests from web application test automation tools (for example, Cypress or Selenium)

Details

Requirements	Users	Best for...
<ul style="list-style-type: none">Automation tests (regression, integration, smoke, or performance tests)A tool like Cypress or Selenium to orchestrate the tests.A server to instrument applications	QA, automation engineers and DevOps developers	All environments

Deployment with API test tools

This method of exercising applications involves deployment to a server that receives requests from API tests tools (for example, Postman).


Details

Requirements	Users	Best for...
A developer with access to Postman in a development environment, or a server and Postman automation in a QA environment.	A developer who instruments a local machine, or a QA engineer who's written a Postman script for QA.	All environments

Deployment with DAST tools

This method of exercising applications involves deploying a server alongside an existing dynamic analysis security testing tools (DAST) such as Rapid7, Veracode, or Netsparker.


Details

Requirements	Users	Best for...
DAST tool needs to be in place. Most likely, this is a tool you need to purchase.	Security and DevOps staff who deploy a built-in application to a server for DAST.	A test environment that uses a DAST tool.
<div>NOTE Consider this method supplementary to other types of testing.</div>		

Deployment with open-source crawlers

This method of exercising applications involves using an open-source crawler (for example, Zap).

Details

Requirements	Users	Environment
<ul style="list-style-type: none">Free crawling toolImplementation in a QA environment or local usage in a development environment.	<ul style="list-style-type: none">DevOps users who integrate open-source crawlers into a CI/CD pipeline.Developers who run the tool manually	Any automated deployment and test environment
<div>NOTE Open-source crawlers tools may not fully exercise the application due to the nature of the payload.</div>		

Deployment with manual penetration testing

This method of exercising applications involves deployment to a server that is used for manual penetration testing environment (in house or third parties like NetSPI).

Details

Requirements	Users	Best for...
<ul style="list-style-type: none">Purchase of pen testing servicesScheduling tests.	In-house or third-party testers	Environments that have in-house penetration testers. If you are using third-party penetration testers, consider using Contrast Assess as a way to reduce costs of pen tests.

Deployment with Burp Suite-based penetration testing

This method of exercising applications involves deployment to a server that is used for Burp Suite based penetration tests.

Details

Requirements	Users	Best for...
Burp Suite to enable BurpTrast integration. The free version is useful because it is easy for security staff to use it. Use the paid version if you previously purchased it.	In-house penetration testers or security staff.	Environments that use the free or paid version of Burp Suite. In most cases, the free version of Burp Suite is sufficient because security staff can use it directly. If you already purchased , use that version.

User curl commands with Assess data

This method of exercising applications involves the use of curl ommands with Assess data.

Details

Requirements	Users	Best for...
<ul style="list-style-type: none"> • Use of a command line interface • Ability to authenticate an API call 	Anyone with access to a server and an authorization token to call specific API	<p>Users who have server and token information.</p> <p>Consider this method supplementary to other types of testing.</p>

Java agent

The Contrast Java agent adds either Contrast Assess or Contrast Protect analysis to Java based applications. The agent analyzes Java web applications built on traditional application servers, and newer Java web applications such as those built with Netty, Play or Spring Boot. If there's a JVM, the Java agent can provide security insights.

As your application runs, the Java agent's sensors gather information about the application's security, architecture and libraries. You can see the results of the agent's analysis in Contrast.

To start analyzing an application, [install the Java agent \(page 123\)](#).

Supported technologies for Java (Kotlin, Scala) agent

Java

Technology	Supported versions	Notes
Java runtime	<ul style="list-style-type: none"> • IBM 8 • Oracle 8. *Versions 11+ follow our OpenJDK support. • OpenJDK 8, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 23 	<p>OpenJDK support is designed to work with all publicly available builds within the current version support shown here. Popular varieties like Azul and Amazon Corretto fall into this category of supported JDKs.</p> <p>Not supported:</p> <p>JDK preview features</p>
Java runtime for Legacy Java agent	<ul style="list-style-type: none"> • IBM 6, 7 • Oracle 6, 7 • OpenJDK 6, 7 	<p>See also</p> <ul style="list-style-type: none"> • Support-Bulletin-End-of-Support-of-Java-6-and-7 • FAQ: End of Support for Java 6 and 7
Use with Java agent 3.x only		
Application servers	<ul style="list-style-type: none"> • GlassFish 4, 5, and 6 • Grizzly 2.3.20 and later • JBoss EAP 6.x and 7.x • Jetty 7, 8, 9, 10, 11 • Karaf 3.0.x • Netty 4.x • Payara 5, 6 • Play 2.4 • Resin 4 • Tomcat 5, 6, 7, 8, 9, 10 • Vert.X 3.1.0, 4.x • WebLogic 10, 11g, 12c, 14 • WebSphere* 8.5, 9.0 • WebSphere Liberty 22, 23, 24 • WildFly 10, 11, 14, 18, 23-32 	<p>* Contrast offers limited support for zSeries and AIX environments. Customers using WebSphere on SPARC Solaris require version 8.5.5.11.</p> <p>Route coverage support:</p> <ul style="list-style-type: none"> • GlassFish 4, 5, and 6 • Jetty 11.0, 10.0. 9.4, 8.1, and 7.6 • Resin 4.0 • Tomcat 5, 6, 7, 8, 9, and 10 • WebLogic 12, 14 • WebSphere 8.5 and 9.0 • Wildfly 10, 11, 14, 18, 23-32

Technology	Supported versions	Notes
Optimizers	Proguard	Proguard includes Java bytecode optimization features which break basic assumptions that runtime agents like Contrast rely on. Proguard users that want to protect their applications with Contrast need to avoid these optimizations by using Proguard's -dontoptimize configuration option.
Databases	<ul style="list-style-type: none"> DB2 DynamoDB MySQL Oracle PostgreSQL SQL Server SQLite JDBC drivers 	
Message services	<ul style="list-style-type: none"> Message services: JMS 2.0 IBM MQ 9.x Spring JMS 2.x 	<ul style="list-style-type: none"> Agent version: Java 4.7.0 and later Contrast version: 3.9.9 and later
Contrast supports message services for Contrast Assess only.	<ul style="list-style-type: none"> Kafka streaming 	<ul style="list-style-type: none"> Agent version: Java 5.0.0 and later
Other Java technologies	<ul style="list-style-type: none"> ADF JSF Apache POI, fileupload, HttpComponents Axis (RPC), XMLRPC, RMI, Apache CXF, JMS (javax.jms) Direct Web Remoting (DWR) DropWizard Freemarker Glowroot* GSON, Kryo, minidev, org.json Google Web Toolkit (GWT) gRPC 1.4.x, 1.5.x, 1.6.x Hibernate 6.2 - 6.4 http4k (4.6.0.0 and 4.17 for Contrast Assess) J2SE JDBC, JDBI, MongoDB JSF (MyFaces, RichFaces, Sun) java.nio, java.beans Java EE/J2EE, Servlet/JSP Jersey MyBatis OWASP ESAPI, AntiSamy, Coverity PrimeFaces Quarkus RESTeasy Seam Spring, Spring Boot, Spring AOP Spring Cloud Spring WebFlux 5 and 6 Spring Web Services Struts, Struts 2 Wicket XStream, Jackson (JSON/XML) Xerces, JAXB, nu.xom 	<p>*If you are using Glowroot, the Contrast Java Agent jar should be included and loaded prior to the Glowroot jar.</p> <p>Route coverage support:</p> <ul style="list-style-type: none"> http4k-core 4.17 http4k-core 4.6 Jersey server 2.25, 2.28, 2.36, 2.6 Quarkus RESTeasy 2.15 Spring Web MVC 4.2, 5.3, and 6.0 Spring WebFlux 5 and 6 Spring Web Services Struts 2

Kotlin

Technology	Supported versions
Contrast agent	3.9.1.25108 and later
Java Run Time	JDK 8 and up

Technology	Supported versions
Kotlin version	1.5.x - 1.8.x

Scala

Technology	Supported versions
Contrast agent	3.8.11.23624 and later
Java Run Time	JDK 8 and up
Scala version	2.12, 2.13
Play version	2.6, 2.7, 2.8
Akka HTTP	10.2.4

WebSphere configuration

If you are using WebSphere as an application server, refer to the information in [Configure the Java agent for Websphere \(page 147\)](#) before you deploy the agent.

Java deduplication for middleware vulnerabilities

Starting with Java 6.11.1, the agent contains enhancements that reduce redundant or duplicate vulnerabilities that occur in applications using middleware components, such as filters. This update improves the accuracy and efficiency of Contrast vulnerability reporting by deduplicating findings that originate from these middleware layers.

Middleware components play an essential role in web applications by intercepting and processing requests before they reach servlets and controllers. They handle responses after servlets or controllers complete execution.

A filter is a component that intercepts requests before they are passed to a servlet and processes responses before they are returned to the client. Filters have access to the request's routing data, current controller, and other contextual information.

Why is this update important?

Often, vulnerabilities tied to both routes and middleware result in duplicated findings, which inflate the number of vulnerabilities that security teams need to review. This situation causes wasted time, missed priorities, and inefficiency in remediation efforts. By enabling the new deduplication capability, Contrast reduces unnecessary noise and improves the overall security posture of applications.

Features

- **Middleware Instrumentation:** The Java agent now recognizes filters and middleware as unique sources, allowing vulnerabilities identified within these layers to be appropriately categorized and reported.

The name of a vulnerability includes an indication that Contrast found it in a middleware component. For example, a vulnerability that Contrast discovers in a filter could look similar to the following example

```
Path Traversal from "param" Parameter in com.contrastsecurity.testapp.servlet24.route.coverage.Filters$ServletMappedFilter.doFilter(javax.servlet.ServletRequest,javax.servlet.ServletResponse,javax.servlet.FilterChain)
```

- **Deduplication Process:** Once the agent detects middleware vulnerabilities, Contrast automatically deduplicates them from the route-based vulnerabilities to which they are attached.
- **Auto-Verification Support:** Vulnerabilities that the agent discovers in middleware are eligible for time-based auto-verification (see [Vulnerability management policies \(page 1124\)](#)). This type of auto-verification streamlines the process of validating resolved vulnerabilities and reduces manual intervention.
Route-based and session-based auto-verification are not currently supported.
- **Comprehensive Reporting:** Contrast displays middleware vulnerabilities in its vulnerability reporting and telemetry, providing clear visibility into the status and lifecycle of these findings.

How it works

- **Filter handling:** The Java agent instruments filters to gather information on vulnerabilities that may be attached to them. This process involves identifying vulnerabilities in the context of the request lifecycle, specifically in the filters where these vulnerabilities originate.
- **Deduplication logic and vulnerability identification:** The agent treats middleware or filters and routes are treated as separate sources of vulnerabilities. The agent distinguishes between vulnerabilities originating from routes versus those originating from middleware components.
- **Auto-verification:** Contrast automatically includes middleware-based vulnerabilities in the auto-verification workflow. The recommended auto-verification method is a time-based auto-verification to ensure comprehensive coverage. [Set an auto-verification vulnerability policies \(page 1127\)](#) describes how to set time-based auto-verification.

Session-based auto-verification(SBAV) and route-based auto-verification (RBAV)[are not currently included in the workflow today.

Upgrade considerations

- **On-premises customers:** Upgrade to Contrast version 3.11.8 or later before you upgrade the Java agent. If you upgrade the agent before upgrading your Contrast installation , you might see new middleware routes and a potentially significant change to your route coverage metrics.
- **Impact on existing vulnerabilities:** Upgrading the Java agent results in enhanced accuracy in vulnerability reporting.

However, existing vulnerabilities that were duplicated before the update remain in the system. If you have not set up any auto-verification policies, consider manually deleting the outdated vulnerabilities. To identify potential duplicates, on the Vulnerabilities page in the Contrast web interface, select the **Group by sink option**.

For additional assistance, contact your Customer Success Manager or Contrast Support.

Install the Java agent

There are several ways to install the Java agent depending on your situation. You might want to consider where you want to use Contrast (for example, Assess in your development environment or Protect in your production environment), your existing build tools, and how your application is deployed.



TIP

If you are using multiple agent-based technologies in parallel with the Contrast Java agent, ensure that you specify the Contrast Java agent as the first agent that loads at startup. For example:

```
java -javaagent:contrast.jar -javaagent:newrelic.jar
```

Loading the Contrast Java agent first helps to limit performance impacts.

Contrast and hot deployments

A hot deployment is the process of adding new components (such as WAR files, servlets, and JSP files) to a running server without having to stop and restart the application server process.

The Contrast agent continues to work during hot deployments and hot reloads with these considerations:

- Contrast might not detect libraries that are added or removed dynamically during hot deployments.

- Contrast cannot update session metadata during a hot deployment.
- Some WebSphere users might experience issues.

If you encounter hot deployment issues, restart the application server.

Contrast and OpenTelemetry agents

If you plan to use an OpenTelemetry agent in the same environment as the Contrast agent, consider suppressing OpenTelemetry instrumentation of Contrast classes. Doing so prevents possible conflicts with Contrast agents.

To suppress OpenTelemetry instrumentation, add this exclusion as an environment variable:

```
OTEL_JAVAAGENT_EXCLUDE_CLASSES="com.contrast*"
```

Or a JVM option:

```
-Dotel.javaagent.exclude-classes=com.contrast*
```

Quick start

Just want to try out the Java agent and see how it works? Check out this [Java Quick Start Guide \(page 148\)](#).

Basic installation

To install the Java agent in most situations (like in an application server like Tomcat, or a container like Docker), choose a repository and follow these instructions to download and install the agent:

- [Maven Central \(page 124\)](#)
- [Debian \(page 125\)](#)
- [RPM \(page 126\)](#)

Build-integrated installation

If you are using Assess in a development environment, and you want to set the build outcome in an existing software project if vulnerabilities are found, install the agent with:

- [Maven plugin \(page 1103\)](#)
- [Gradle plugin \(page 1084\)](#)
- [Jenkins plugin \(page 1089\)](#)

Install the Java agent using Maven Central

The Contrast Java agent is [available from Maven Central](#) using group ID `com.contrastsecurity` and artifact ID `contrast-agent`. To install the Java agent:

1. Get the *contrast-agent.jar* from Maven Central. ([See examples](#) of how to download from the Maven repository.)
The latest Contrast Java agent is available for download directly from <https://download.java.contrastsecurity.com/latest>.
2. [Configure the agent \(page 149\)](#). You can create or download a [YAML configuration file \(page 107\)](#). You must provide Contrast connection parameters using [these agent keys \(page 104\)](#).
3. Tell the agent where to find the yaml configuration file (`contrast.yaml`). In the example below, substitute `<YourContrastJarPath>` with the path to your Contrast JAR (this may vary depending on your internal file structure and how you downloaded the file) and `<ApplicationJar>` with the name of your application JAR.

```
java -javaagent:<YourContrastJarPath>  
-Dcontrast.config.path=contrast.yaml -jar <ApplicationJar>.jar
```


**NOTE**

If you are using system properties, environment variables to configure instead of YAML, or you have placed the YAML in a [standard location \(page 106\)](#) where the agent can find it automatically, set the JVM parameter to include the Java agent.

```
java -javaagent:<YourContrastJarPath> -jar <AppName>.jar
```

4. Use the application as you normally would (for example, click on the web interface, send API commands). Verify that Contrast sees your application (for example, view your application in the Contrast web interface, view logs).

Contrast artifacts deployed to Maven Central are signed with our GPG key hosted on <https://keyserver.ubuntu.com>. Contrast's public signing key has ID 1AAD9AFB3FC5CCA6940D021534D84B137E8F1053 and can be installed to a local keyring with the following command:

```
gpg --keyserver keyserver.ubuntu.com --recv-keys  
1AAD9AFB3FC5CCA6940D021534D84B137E8F1053
```

You can also provide security analysis for applications running in a test/QA or production environment, by installing the agent with an application server like:

- [Jetty \(page 144\)](#)
- [JBoss/Wildfly \(page 143\)](#)
- [Tomcat \(page 145\)](#)
- [WebLogic \(page 146\)](#)
- [WebSphere \(page 147\)](#)

You can also [install using a container \(page 127\)](#), like Docker.

**TIP**

Check the [Contrast Support Portal](#) for more information about other compatible ways to install the agent. If you are using VMware Tanzu, see the details in [Java installation with VMware Tanzu \(page 132\)](#).

Install the Java agent using the Debian repository

You can configure your system to retrieve and install the Java agent from the Contrast Debian repository. To do this:

1. Use the following commands to configure your system to receive packages from the repository:

```
curl https://pkg.contrastsecurity.com/api/gpg/key/public | sudo apt-key  
add -  
echo "deb https://pkg.contrastsecurity.com/debian-public/ all contrast"  
| sudo tee /etc/apt/sources.list.d/contrast-all.list
```

2. Install the Contrast Java agent:

```
sudo apt-get update && sudo apt-get install contrast-java-agent
```

3. You will now see the Contrast Java agent JAR file at `/opt/contrast/contrast-agent.jar`.

4. [Configure the agent \(page 149\)](#). You can create or download a [YAML configuration file \(page 107\)](#). You must provide Contrast connection parameters using [these agent keys \(page 104\)](#).
5. Tell the agent where to find the yaml configuration file (`contrast.yaml`). In the example below, substitute `<YourContrastJarPath>` with the path to your Contrast JAR (this may vary depending on your internal file structure and how you downloaded the file) and `<ApplicationJar>` with the name of your application JAR.

```
java -javaagent:<YourContrastJarPath>  
-Dcontrast.config.path=contrast.yaml -jar <ApplicationJar>.jar
```



NOTE

If you are using system properties, environment variables to configure instead of YAML, or you have placed the YAML in a [standard location \(page 106\)](#) where the agent can find it automatically, set the JVM parameter to include the Java agent.

```
java -javaagent:<YourContrastJarPath> -jar <AppName>.jar
```

6. Use the application as you normally would (for example, click on the web interface, send API commands). Verify that Contrast sees your application (for example, view your application in the Contrast web interface, view logs).

You can also provide security analysis for applications running in a test/QA or production environment, by installing the agent with an application server like:

- [Glassfish](#)
- [Jetty \(page 144\)](#)
- [JBoss/Wildfly \(page 143\)](#)
- [Tomcat \(page 145\)](#)
- [WebLogic \(page 146\)](#)
- [WebSphere \(page 147\)](#)

You can also [install using a container \(page 127\)](#), like Docker.



TIP

Check the [Contrast Support Portal](#) for more information about other compatible ways to install the agent. If you are using VMware Tanzu, see the details in [Java installation with VMware Tanzu \(page 132\)](#).

Install the Java agent with the RPM repository

To install the Java agent with the RPM repository:

1. Use the following commands to configure your system to retrieve packages from the Contrast RPM repository:

```
OSREL=$(rpm -E "%{rhel}")  
sudo -E tee /etc/yum.repos.d/contrast.repo << EOF  
[contrast]  
name=contrast repo  
baseurl=https://pkg.contrastsecurity.com/rpm-public/centos-$OSREL/
```

```
gpgcheck=0
enabled=1
EOF
```

2. Once you've finished configuration, install the Contrast Java agent:

```
sudo yum install contrast-java-agent
```

3. The Contrast Java agent JAR is now installed at `/opt/contrast/contrast-agent.jar`.
4. [Configure the agent \(page 149\)](#). You can create or download a [YAML configuration file \(page 107\)](#). You must provide Contrast connection parameters using [these agent keys \(page 104\)](#).
5. Tell the agent where to find the yaml configuration file (`contrast.yaml`). In the example below, substitute `<YourContrastJarPath>` with the path to your Contrast JAR (this may vary depending on your internal file structure and how you downloaded the file) and `<ApplicationJar>` with the name of your application JAR.

```
java -javaagent:<YourContrastJarPath>
-Dcontrast.config.path=contrast.yaml -jar <ApplicationJar>.jar
```



NOTE

If you are using system properties, environment variables to configure instead of YAML, or you have placed the YAML in a [standard location \(page 106\)](#) where the agent can find it automatically, set the JVM parameter to include the Java agent.

```
java -javaagent:<YourContrastJarPath> -jar <AppName>.jar
```

6. Use the application as you normally would (for example, click on the web interface, send API commands). Verify that Contrast sees your application (for example, view your application in the Contrast web interface, view logs).

You can also provide security analysis for applications running in a test/QA or production environment, by installing the agent with an application server like:

- [Glassfish](#)
- [Jetty \(page 144\)](#)
- [JBoss/Wildfly \(page 143\)](#)
- [Tomcat \(page 145\)](#)
- [WebLogic \(page 146\)](#)
- [WebSphere \(page 147\)](#)

You can also [install using a container \(page 127\)](#), like Docker.



TIP

Check the [Contrast Support Portal](#) for more information about other compatible ways to install the agent. If you are using VMware Tanzu, see the details in [Java installation with VMware Tanzu \(page 132\)](#).

Install the Java agent using a container

This topic provides general guidance for installing the Contrast Java agent in a containerized application, with Docker as an example.

**NOTE**

If the agent takes a long time to start, [Java Agent Effects on Startup Performance](#) and [Java agent with Docker](#) provide details to help you resolve this issue.

Before you begin

You should have a basic understanding of how containers and related software work. You may need to adjust the instructions to meet your specific circumstances.

ECS support

Using this procedure, you can install the Contrast Java agent using a Docker container in an Amazon Elastic Container Service (ECS) environment.

Step 1: Install the agent

Contrast can be added either before or after the application is added to the container image. The recommended approach is with the use of named [multi-stage builds](#). For example:

```
FROM eclipse-temurin:17

# Hidden for brevity...

# Copy the required agent files from the official Contrast agent image.
COPY --from=contrast/agent-java:latest /contrast/contrast-agent.jar /opt/contrast/contrast.jar
```

In this example, the latest Java agent is used. Check DockerHub for [available tags](#).

Step 2: Configure the agent

When installing the Java agent into a container:

- Use a **YAML configuration file** for common configuration settings so it can be placed in the base image. For example, a common configuration might include redirecting logging to console output, proxy configuration, or performance tuning. The [Contrast agent configuration editor \(page 109\)](#) can help with configuring the agent correctly. Create and copy the YAML file into the base image, then copy the file into the base image Dockerfile using:

```
COPY WORKSPACE/contrast_security.yaml /opt/contrast/contrast_security.yaml
```

- Use **Java system properties** or **environment variables** for application-specific configuration values so you can uniquely configure options for each application.

Contrast configuration	Function	Java system property	E
Application metadata	Specify application-specific metadata Create application metadata (page 1174) before you specify them in the configuration.	-Dcontrast.application.metadata	CONTRAST__AP

Contrast configuration	Function	Java system property	Environment variable
Application session metadata	Send application details like build number, version, GIT hash, and other session metadata (page 616) .	<code>-Dcontrast.application.session_metadata</code>	<code>CONTRAST__AP</code>
Application group	Specify the application access group for this application during onboarding. Create these groups (page 1164) in Contrast first.	<code>-Dcontrast.application.group</code>	<code>CONTRAST__AP</code>
Server environment	Specify in which environments the application is running: Development, QA and Production.	<code>-Dcontrast.server.environment</code>	<code>CONTRAST__SE</code>

Step 3: Update JVM parameters

To attach any profiler to a Java application, you need to pass a `-javaagent` flag to the application by setting `JAVA_TOOL_OPTIONS` environment variables.

Pre-populate the Contrast common JVM parameters in a separate environment variable in the base image, so the application team can use it in `JAVA_TOOL_OPTIONS`. For example:

- For the base image Dockerfile:

```
ENV CONTRAST_OPTS "-javaagent:/opt/contrast/contrast.jar \
-Dcontrast.config.path=/opt/contrast/contrast_security.yaml"
```

- For the application image Dockerfile:

```
ENV JAVA_TOOL_OPTIONS $CONTRAST_OPTS \
-Dcontrast.application.metadata=bU=<value>,contactEmail=<value>,contactNam
e=<value> \
-Dcontrast.application.group=APP_GROUP
```

Step 4: Run the application image

After you [add \(page 128\)](#) and [configure \(page 128\)](#) the agent in a base image, run the image.

For the agent to send data to Contrast, it needs [agent authentication keys \(page 104\)](#). To protect the agent credentials, you can use the Docker secret and pass them as environment variables during deployment time. Here is an example of the Docker run command:

```
docker run -e CONTRAST__API__URL=https://app.contrastsecurity.com -e
CONTRAST__API__API_KEY=<value> -e CONTRAST__API__SERVICE_KEY=<value> -e
CONTRAST__API__USER_NAME=<value> -e CONTRAST__SERVER__NAME=<value> -e
CONTRAST__SERVER__ENVIRONMENT=<value> image_with_contrast
```

You can verify that Contrast is running by checking the container log. You should see messages like these:

```
2020-05-28 22:36:29,910 [main STDOUT] INFO - Copyright: 2019 Contrast
Security, Inc
2020-05-28 22:36:29,910 [main STDOUT] INFO - Contact:
support@contrastsecurity.com
2020-05-28 22:36:29,910 [main STDOUT] INFO - License: Commercial
```

```
2020-05-28 22:36:29,910 [main STDOUT] INFO - NOTICE: This Software and the
patented inventions embodied within may only be used as part of
2020-05-28 22:36:29,910 [main STDOUT] INFO - Contrast Security's commercial
offerings. Even though it is made available through public
2020-05-28 22:36:29,910 [main STDOUT] INFO - repositories, use of this
Software is subject to the applicable End User Licensing Agreement
2020-05-28 22:36:29,910 [main STDOUT] INFO - found at https://
www.contrastsecurity.com/enduser-terms-0317a or as otherwise agreed between
2020-05-28 22:36:29,910 [main STDOUT] INFO - Contrast Security and the End
User. The Software may not be reverse engineered, modified,
2020-05-28 22:36:29,910 [main STDOUT] INFO - repackaged, sold,
redistributed or otherwise used in a way not consistent with the End User
2020-05-28 22:36:29,910 [main STDOUT] INFO - License Agreement.
[Contrast] Thu May 28 22:36:30 EDT 2020 Effective instructions:
Assess=false, Protect=true
[Contrast] Thu May 28 22:36:30 EDT 2020 String Supporter has been disabled
[Contrast] Thu May 28 22:36:30 EDT 2020 Logging security messages to /Users/
usernamehere/.contrast/security.log
[Contrast] Thu May 28 22:36:31 EDT 2020 Starting JVM [1862ms]
```

See also

[Agent Operator \(Kubernetes operator\) \(page 552\)](#)

Contrast Support Portal [AWS Fargate and Contrast agents](#) and [Java agent with Docker](#)

Install Java with infrastructure as code tools

If you are using Ansible or Chef in your deployment environments, you can use either of them to deploy the Java agent.

[Ansible playbook \(page 94\)](#)

[Chef cookbook \(page 62\)](#)

Install the Java agent in an existing Gradle project with Docker

This example uses a sample Gradle project, which includes the [Application Plugin](#) and the [Docker Plugin](#) to build a Java web application. It also runs JUnit 5 integration tests that verify the web application's behavior. As part of the process, you will include Contrast in the Docker image used for testing so that Contrast Assess analyzes your code during integration testing. See an [example of a Gradle project](#) in our Github repo.



NOTE

Any part of the following procedures that refer to any form of packaging or distribution are meant for your organization's internal use. Do not distribute Contrast with your application or Docker container outside of your organization. See [Contrast's Terms of Service agreement](#) for more information.

To add the Contrast Java agent to an existing Gradle project with Docker:

1. Open a command prompt, and run the following command to clone Contrast's **examples** repository:

```
$ git clone https://github.com/Contrast-Security-OSS/contrast-java-examples.git
```

2. Enter the `gradle-docker` directory:

```
$ cd contrast-java-examples/gradle-docker
```

3. Run a test build to make sure everything is working:

```
$ ./gradlew build
```

```
BUILD SUCCESSFUL in 3s
```

```
4 actionable tasks: 3 executed, 1 up-to-date
```

**NOTE**

On Windows, run `gradlew.bat build` instead.

4. If the test build doesn't work, check to make sure you have Java 11 correctly installed (Java 11 or later is required to build the sample application. [Java supported technologies \(page 120\)](#) lists the versions of Java supported that the Contrast Java agent supports):

```
$ java -version
openjdk version "11.0.18" 2023-01-17
OpenJDK Runtime Environment Temurin-11.0.18+10 (build 11.0.18+10)
OpenJDK 64-Bit Server VM Temurin-11.0.18+10 (build 11.0.18+10, mixed mode)
```

5. If you've made changes, run the build again. If it still doesn't work, [open an issue](#) that explains the problem.
6. Use the [agent keys \(page 104\)](#) to configure the agent's communication with Contrast. In most cases, you'll use this key:

- **Agent Token:** This variable is a base64 encoded JSON object containing the `url`, `api_key`, `service_key`, and `user_name` configuration settings, allowing you to set them set in a single variable.

Legacy settings: For older agents, you'll need these keys:

- **Contrast URL:** This URL, `https://app.contrastsecurity.com/Contrast` or the URL of your on-premises or private cloud instance.
 - Organization API key
 - Agent username
 - Agent service key
7. Add the key as Gradle properties to the `gradle.properties` file in your [Gradle user home directory](#). If this file does not exist, create it.

```
contrastToken=<contrast_api_token>
```

Legacy settings: If you are using an older agent version, add these keys to the `gradle.properties` file.

Be sure to replace `<contrast_url>`, `<your_api_key>`, `<agent_user_name>` and `<agent_user_service_key>` with the Contrast URL, API key, username and service key values you obtained from the Contrast:

```
contrastUrl=<contrast_url>
contrastAgentUserName=<agent_user_name>
contrastAgentServiceKey=<agent_user_service_key>
contrastApiKey=<your_api_key>
```

8. Add the Contrast agent and configure the application to use it by modifying the `createDockerfile` task in `build.gradle`:

```
task createDockerfile(type: Dockerfile) {  
    // ... rest of block omitted  
  
    copyFile(new Dockerfile.CopyFile("/contrast/contrast-agent.jar", "/  
contrast.jar").withStage("contrast/agent-java:latest"))  
    environmentVariable("JAVA_TOOL_OPTIONS", "-javaagent:/contrast.jar")  
}
```

9. Pass the configuration variables into the container by adding the following commands to the `createContainer` task in `build.gradle`:

```
task createContainer(type: DockerCreateContainer) {  
    // ... rest of the config omitted  
  
    envVars = [  
        CONTRAST__API__TOKEN: project.property("contrastToken"),  
        CONTRAST__APPLICATION__NAME: "${project.name}-how-to"  
    ]  
}
```

If your agent configuration refers to both the legacy settings and the agent token, (in environment variables or the YAML file), the legacy settings take precedence. To use just the agent token value, make sure you remove references to the legacy settings

Legacy settings: If you are using an older agent version, add the following commands:

```
task createContainer(type: DockerCreateContainer) {  
    // ... rest of the config omitted  
  
    envVars = [  
        CONTRAST__API__URL: project.property("contrastUrl"),  
        CONTRAST__API__USER_NAME:  
project.property("contrastAgentUserName"),  
        CONTRAST__API__SERVICE_KEY:  
project.property("contrastAgentServiceKey"),  
        CONTRAST__API__API_KEY: project.property("contrastApiKey"),  
        CONTRAST__APPLICATION__NAME: "${project.name}-how-to"  
    ]  
}
```

10. Run the build again:

```
./gradlew clean build
```



NOTE

On Windows, run `gradlew.bat clean build` instead.

The Docker container now runs the application with Contrast enabled. When the integration test runs, it detects the vulnerable endpoint and reports it to Contrast. To see the vulnerability report, log in to the Contrast web interface, navigate to the [Vulnerabilities list \(page 1022\)](#) and filter your view by the application name **gradle-application-how-to**.

Java agent installation with VMware Tanzu Application Service

VMware Tanzu (Application Service formerly Pivotal Cloud Foundry) is a proprietary containerized Software as a Service (SaaS) environment. A Java buildpack that VMware releases makes the Contrast Java agent accessible. You install the buildpack in the container where you run your Java application.

Contrast service

The existence of a single, bound Contrast service activates and downloads the Java agent. The `VCAP_SERVICES` payload, containing a service name, label or tag with `contrast-security` as a substring, defines the Contrast service. You can use either of these methods to create the Contrast service:

- **User-provided service:** A user-provided service is a simple way to bind a single application to the Java agent and configure authentication.
- **A service broker (Contrast tile):** Use the service broker to bind multiple applications, providing access to the Java agent and authentication.

When the Contrast service is bound to your application, it provides the strings needed to activate Contrast (puts the `javaagent` flag in the JVM) and provides authentication to the Contrast web interface.

Java buildpacks

Java buildpacks contain the instructions and configuration information that the container needs to download and configure the Java agent. You can use an offline or online buildpack:

- An offline buildpack is typically forked from the GitHub repo where you have made customizations. These repos might contain older agent versions.
- An online buildpack is usually the latest version and pulled from GitHub when needed.

Requirements

- Buildpacks
To instrument an application in a VMware Tanzu Network environment, your application must use one of these buildpacks:
 - [Cloud Foundry Java Buildpack](#), version 3.19 and later or version 4.2 and later
 - [IBM Liberty Buildpack](#), version 2.7.0.2 and later
- Name or tag with `contrast-security` specified when you create the service
- The credential payload must contain the standard YAML properties.

For general information on configuring the buildpack, including how to specify configuration values through environment variables, refer to the [Configuration and Extension](#) section of the Cloud Foundry Java Buildpack documentation.

Configuration options

You can configure the framework by modifying the `config/contrast_security_agent.yml` file in the buildpack fork. The framework uses the [Repository utility support](#) and supports the [version syntax](#) defined there.

Name	Description
<code>repository_root</code>	The URL of the Contrast Security repository index
<code>version</code>	The version of the Contrast agent to use

To specify a version of the Java agent to use, set the `JBP_CONFIG_CONTRASTSECURITYAGENT` environment variable and specify a version listed in the [index](#). For example:

```
JBP_CONFIG_CONTRASTSECURITYAGENT='version: 4.13.1'
```

Example

This example shows how to create a user-provided service and bind it to an application called `spring-petclinic`:

1. This command pushes an application to Cloud Foundry, providing the buildpack to be used: (otherwise the default buildpack is used in the environment)

```
cf push myApp -p target/spring-petclinic-2.4.2.jar \
  -b 'https://github.com/cloudfoundry/java-buildpack.git'
```

2. This command creates a user-provided service:

```
cf create-user-provided-service contrast-security-service -p
"teamserver_url, username, api_key, service_key"
```

The value for `teamserver_url` should include only the protocol and hostname. Do not include `/contrast/` or `/contrast/api`.

3. This command binds the service to the application (this task is essential):

```
cf bind-service myApp contrast-security-service
```

4. This command restages the application so it can connect to Contrast (essentially, restarts the container):

```
cf restage myApp
```

See also

[Add Contrast service broker tile for VMware Tanzu \(page 136\)](#)

[Add Contrast service broker for VMware Tanzu \(page 134\)](#)

[Configure a proxy for Contrast service broker \(page 137\)](#)

Add Contrast service broker for VMware Tanzu

Steps

1. Deploy the service broker application with a command similar to this example:

```
cf push contrast-security-service-broker
```

You should see the service broker in PCF.

2. Configure plans with the `CONTRAST_SERVICE_PLANS` environment variable (the service broker doesn't offer any plans by default).

You can also use the Pivotal Ops Manager to set the environment variables. If you are using IBM Cloud, you can select the application, select **Runtime** and then **Environment Variables** to set the value.

Example: This example shows how to set the value in the command line:

```
cf set-env contrast-security-service-broker CONTRAST_SERVICE_PLANS
" {
  "ServicePlan1": {
    "name": "ServicePlan1",
    "teamserver_url": "https://yourteamserverurl.com",
    "username": "your_username",
    "org_uuid": "00000000-1111-2222-3333-000000000000",
    "api_key": "your_api_key",
    "service_key": "your_service_key"
  },
  "AnotherServicePlan": {
    "name": "AnotherServicePlan",
    "teamserver_url": "https://yourteamserverurl.com",
    "username": "your_username",
```

```

        "org_uuid": "00000000-1111-2222-3333-000000000001",
        "api_key": "your_api_key",
        "service_key": "some_other_service_key"
      }
    } "

```

To run the agent on IBM Cloud, you must use single quotes to set the `CONTRAST_SERVICE_PLANS` environment variable, as shown in this example:

```

cf set-env contrast-security-service-broker CONTRAST_SERVICE_PLANS
" {
  'ServicePlan1': {
    'name': 'ServicePlan1',
    'teamserver_url': 'https://yourteamserverurl.com',
    'username': 'your_username',
    'org_uuid': '00000000-1111-2222-3333-000000000000',
    'api_key': 'your_api_key',
    'service_key': 'your_service_key'
  },
  'AnotherServicePlan': {
    'name': 'AnotherServicePlan',
    'teamserver_url': 'https://yourteamserverurl.com',
    'username': 'your_username',
    'org_uuid': '00000000-1111-2222-3333-000000000000',
    'api_key': 'your_api_key',
    'service_key': 'some_other_service_key'
  }
} "

```

3. Restage your application using a command similar to this example:

```
cf restage contrast-security-service-broker
```

4. Set an environment variable for a username and a password:

```

cf set-env contrast-security-service-broker SECURITY_USER_NAME
aSecureUsername
cf set-env contrast-security-service-broker SECURITY_USER_PASSWORD
aSecurePassword

```

5. Create a service broker instance. Define at least one service plan for this. You must use the same username and password that you set in the previous step.

```

cf create-service-broker contrast-security-service-broker USER_NAME
PASSWORD
<URL of your application>

```

For IBM Cloud, add `--space-scoped` at the end of the command, as shown in this example:

```

cf create-service-broker contrast-security-service-broker USER_NAME
PASSWORD
<URL of your application> --space-scoped

```

6. All service brokers start as private. Make it public with a command similar to the following example:

```
cf enable-service-access contrast-security-service-broker
```

7. Once the service broker is working, create a service instance and bind it to the application. To create a service instance, run the following command:

```

cf create-service contrast-security-service-broker ServicePlan1
<name_of_service>

```

8. Bind the service broker to your application using the following command:

```
cf bind-service <app_name> <name_of_service>
```

You should see the agent start up with your application. You also see your application in the Contrast web interface.

See also

[Add Contrast service broker tile \(page 136\)](#)

[Configure a proxy for Contrast service broker \(page 137\)](#)

Add the Contrast service broker tile for VMware Tanzu

To integrate Contrast with VMware Tanzu Network (formerly Pivotal Cloud Foundry), install the Contrast service broker tile.

Steps

1. Download the Contrast service broker tile from [VMware Tanzu Network](#).
2. Store the file locally and navigate to your Pivotal Ops Manager instance.
3. Select **Import a Product** and then, select the `contrast-security-service-broker-#.#.#.pivotal` tile that you downloaded.
If the file you downloaded has a ZIP extension, rename it to `contrast-security-service-broker-#.#.#.pivotal`.
4. The tile requires some configuration before you can deploy it. The service broker does not include service plans by default. Add at least one plan before you deploy the Contrast service broker tile. To add a service plan, select **Service Plans** in the Contrast service broker tile and select **Add**.
5. Set these configuration parameters in the service plan:
 - **TeamServer:** The URL for your Contrast application instance
 - **TeamServer Service Key:** [Organization service key \(page 104\)](#)
 - **TeamServer API Key:** [Organization API key \(page 104\)](#)
 - **Organization UUID:** [Organization ID \(page 104\)](#) to which the application will belong
 - **Username:** Your Contrast username
 - **Plan Name:** Name of the plan as it will appear in Apps Manager
 - **Proxy Host:** The hostname of a proxy for the service broker to communicate with Contrast
 - **Proxy Port:** The proxy port
 - **Proxy Username:** The proxy username, if it requires authentication
 - **Proxy Password:** The proxy password, if it required authentication



NOTE

In addition to the proxy settings for the tile, you also need to set up the [agent communication with the proxy. \(page 137\)](#)

6. Select **Save**.
If you want some applications to belong to different organizations, define the other plans you will need.
7. In the dashboard, select **Apply Changes**.
This process can take some time to finish.
8. After you successfully deploy the service broker, you can bind the credentials to an application. Go to the Marketplace to find the Contrast service broker option.
9. In the Pivotal Ops Manager, select the Contrast service broker option to see the available plans that you created.

10. To choose the plan you want to bind to an application, use **Select this Plan**.
11. Specify an instance name for the plan.
This selection doesn't affect the service broker. You can use any name you want for the instance.
12. In the Bind to App drop-down, select the application to bind to this service. Then, restage the application.
This action retrieves the latest agent from Contrast to instrument your application.
13. **Optional:** If you want to override agent properties, such as the application name, set environment variables in PCF using a command similar to the following example:

```
cf set-env APP_NAME JAVA_OPTS "-Dcontrast.agent.java.standalone_app_name=PivotalSpringApp"
```

See also

[Add Contrast service broker \(page 134\)](#)

[Configure a proxy for Contrast service broker \(page 137\)](#)

Set up agent proxy communication for Contrast service broker

If you are deploying a Java agent in VMWare Tanzu, you can choose to configure a proxy when you add the [Contrast service broker tile \(page 136\)](#). The Contrast service broker can use a proxy configuration set within the service plan to complete the binding with Contrast..

If you configure a proxy when you add the Contrast service broker tile, you also need to set up agent communication with the proxy, as described in this topic. You can do this for each application or set it at an organization level for all deployed applications to consume.

Steps

1. To set up proxy communication for each application, use this command:

```
cf set-env $APP_NAME CONTRAST__API__PROXY__ENABLE "true"
cf set-env $APP_NAME CONTRAST__API__PROXY__URL "scheme://host:port"
```

Alternatively, you can use this command:

```
cf set-env $APP_NAME JAVA_OPTS "-Dcontrast.api.proxy.enable=true
-Dcontrast.api.proxy.url=scheme://host:port"
```

2. To set up proxy communication at an organization level, use this command:

```
cf ssevg '{"CONTRAST__API__PROXY__ENABLE": "true"}'
cf ssevg '{"CONTRAST__API__PROXY__URL": "scheme://host:port"}'
```

See also

[Add Contrast service broker tile \(page 136\)](#)

[Add Contrast service broker \(page 134\)](#)

Install the Java agent with AWS Elastic Beanstalk

Use this procedure as a guide to configuring the Java agent to work with AWS Elastic Beanstalk. It describes how to create an `.ebextensions` file that downloads the Contrast Java agent and instruments your application.

Depending on your environment, you might need to customize the steps in this procedure.

This procedure is designed for users who are familiar with DevOps practices and how Beanstalk deployment works.

Before you begin

- Verify that Contrast supports your [preferred tools and environments for Java](#). (page 120)
- Get the information needed to [connect the Java agent to Contrast](#) (page 123).
- Download and start the Contrast Java agent before running your applications.
- Verify that you have access to the Beanstalk environment to install customized `.ebextensions` configuration files.

Step1: Specify settings to download the Contrast Java agent

The `.ebextensions` configuration file has a `files` section that downloads the agent from a remote URL. This example shows how to specify downloading the agent from a Maven repository.

```
files:
  "/opt/contrast/contrast.jar":
    mode: "000755"
    owner: rootCorporate rule
    group: root
    source: "https://repository.sonatype.org/service/local/artifact/maven/
redirect?r=central-proxy&g=com.contrastsecurity&a=contrast-agent&v=LATEST"
```

For Contrast agents, the recommended location is `/opt/contrast`, but you can use another location, if necessary. You can also change the URL to download agents from an internal repository. At build time, you can specify the agent version of your choice and download it from the [Maven repository](#).

Step 2: Create an agent configuration file

There are different values you can use to configure Contrast agents, based on an [order of precedence](#) (page 106). Active configuration values are determined in this order:

1. Corporate rule (for example, expired licenses)
2. System property
3. Environment variable
4. YAML configuration file
5. Contrast web interface value
6. Contrast Security default value

The recommended approach to creating the configuration file is to use a common configuration and an application-specific configuration:

- **Common configuration:** Specifies core set of configurations in the YAML. For example:
 - Redirect logging to console output
 - Proxy configuration, if any
 - Performance tuning options to limit agent activity

This example shows how to create and configure the agent's YAML file at deployment time in an `.ebextensions` configuration file.

```
files:
  "/var/contrast/contrast_security.yaml" :
    mode: "000755"
    owner: root
    group: root
    content: |
      api:
        proxy:
          url: https://host:port
```

```
agent:
  java:
    scan_all_classes: false
    scan_all_code_sources: false
  logger:
    stdout: true
```

- **Application-specific configuration:** This configuration lets you specify additional options, for each application. Use these environment variables:

- **Application metadata:** Specifies application-specific metadata

```
CONTRAST__APPLICATION__METADATA
```

- **Application name:** Specifies the application name reported to Contrast

```
CONTRAST__APPLICATION__NAME
```

- **Application session metadata:** Send application details such as, build number, version, and GIT hash,

```
CONTRAST__APPLICATION__SESSION__METADATA
```



NOTE

Learn about additional [session metadata options](#). (page 618)

- **Application group:** Specifies the application access group for this application when you add it to Contrast. You must create application access groups before you use this variable.

```
CONTRAST__APPLICATION__GROUP
```

- **Server environment:** specify in which environments the application is running. Valid values for this configuration are: Development, QA and Production.

```
CONTRAST__SERVER__ENVIRONMENT
```

Example 1: This example shows how to set environment variables when you create the environment:

```
eb create <environment name> --envvars CONTRAST__API__URL=https://
app.contrastsecurity.com/
Contrast,CONTRAST__API__API_KEY=<value>,CONTRAST__API__SERVICE_KEY=<value>
,CONTRAST__API__USER_NAME=<value>,CONTRAST__SERVER__NAME=<value>,CONTRAST__
_SERVER__ENVIRONMENT=<value>
```

Example 2: This example shows how to set the environment variables after you create the environment:

```
eb setenv CONTRAST__API__URL=https://app.contrastsecurity.com/Contrast
CONTRAST__API__API_KEY=<value> CONTRAST__API__SERVICE_KEY=<value>
CONTRAST__API__USER_NAME=<value> CONTRAST__SERVER__NAME=<value>
CONTRAST__SERVER__ENVIRONMENT=<value>
```

Step 3: Update JVM parameters

To attach any profiler to a Java application, you must pass a `-javaagent` flag to the application. To do this, set the `JAVA_TOOL_OPTIONS` environment variable.

Set these variables in the same way as you set application-specific environment variables. Use the paths for the agent's JAR and YAML configuration files, as shown in this example.

```
eb setenv JAVA_TOOL_OPTIONS="-javaagent:/opt/contrast/contrast.jar
-Dcontrast.config.path=/var/contrast/contrast_security.yaml"
```

Step 4: Deploy the agent using the `.ebextensions` configuration

AWS expects the Beanstalk customization configuration to be in the `.ebextensions` folder in the deployment folder root. This example shows a directory structure that includes the `.ebextensions` folder. It shows the location of the `contrast.config` file that includes the agent download and YAML configuration sections.

```
.ebextensions
  contrast.config
  application.jar
```

Install the Java agent with automatic updates on Linux

Some users like to automatically update their Contrast Java agent software to the latest version. Linux users can schedule Java agent updates from Maven Central using common Linux tools `cron` and `curl`.

Here's how to configure a scheduled Java agent update job on an Ubuntu 18.04 Linux host:



NOTE

Use your preferred editor to create a file with the following contents. The examples provided use `tee` to create the file.

1. If you want to perform each step as you follow along with this guide, you can use [Vagrant](#) and [VirtualBox](#) to create a new Ubuntu 18.04 virtual machine:

```
vagrant init ubuntu/bionic64
```

```
vagrant up
```

```
vagrant ssh
```

2. Create a shared directory for Contrast software:

```
sudo mkdir -p /opt/contrast
```

3. Create a script for installing the latest Java agent in the `/etc/cron.daily` directory. Scripts in this directory execute once daily; as a result, the host updates the latest Java agent each day.
4. Use `tee` to create this script. Press `CTRL+D` when you've finished typing all the lines:

```
$ sudo tee -a /etc/cron.daily/install-latest-contrast-agent > /dev/null
#!/bin/bash -u

CONTRAST_DIRECTORY=/opt/contrast
CONTRAST_FILE_NAME=contrast-agent.jar

CONTRAST_VERSION=$(curl --fail --silent 'https://search.maven.org/
solrsearch/select?q=g:com.contrastsecurity+a:contrast-agent' | sed -e 's/
[{}]'/'/'/g' | sed s/\\"/"/g | awk -v RS=',' -F: '$1=="latestVersion"{print
$2}' | grep -v -e '^$')
curl --fail --silent --
location "https://repo1.maven.org/maven2/com/contrastsecurity/contrast-
agent/${CONTRAST_VERSION}/contrast-agent-${CONTRAST_VERSION}.jar" -o
"contrast-agent-${CONTRAST_VERSION}.jar"
if [ $? -ne 0 ]; then
    echo "Failed to download Contrast Java agent" >&2
```



```
    exit 1
fi
mv /tmp/$CONTRAST_FILE_NAME $CONTRAST_DIRECTORY/$CONTRAST_FILE_NAME
```

5. Set the execute bit on the new script file:

```
sudo chmod +x /etc/cron.daily/install-latest-contrast-agent
```

6. To test the script, execute it and then verify that the file exists using `stat`:

```
$ sudo /etc/cron.daily/install-latest-contrast-agent
$ stat /opt/contrast/contrast-agent.jar
stat /opt/contrast/contrast-agent.jar
  File: /opt/contrast/contrast-agent.jar
  Size: 10568283      Blocks: 20648      IO Block: 4096   regular file
Device: 801h/2049d   Inode: 256034      Links: 1
Access: (0644/-rw-r--r--)  Uid: (   0/   root)   Gid: (   0/   root)
Access: 2019-04-11 02:02:01.265775928 +0000
Modify: 2019-04-11 02:24:47.849796936 +0000
Change: 2019-04-11 02:24:47.849796936 +0000
 Birth: -
```

7. The Contrast agent requires some configuration to communicate with Contrast. You can [find agent key information here \(page 104\)](#).
8. When Contrast is installed on a Linux host, users typically want Contrast-enabled web applications on the host to share basic configuration parameters, such as the ones required to connect to Contrast. By convention, Contrast look for configuration in a YAML file at path `/etc/contrast/java/contrast_security.yaml` on Linux hosts.
9. Create the `/etc/contrast/java` directory:

```
sudo mkdir -p /etc/contrast/java
```

10. Use `tee` to create the configuration file. Replace `<contrast_url>`, `<your_api_key>`, `<agent_user_name>` and `<agent_user_service_key>` with the values you obtained from Contrast in the previous step:

```
$ sudo tee -a /etc/contrast/java/contrast_security.yaml > /dev/null
api:
  url: <contrast_url>
  api_key: <your_api_key>
  user_name: <agent_user_name>
  service_key: <agent_user_service_key>
```

11. Press `CTRL+D` when you've finished typing all the lines.
12. Run a diagnostic test to verify that Contrast is installed and properly configured. The host must have Java installed to execute the diagnostic test:

```
sudo apt install --yes openjdk-11-jre-headless
```

13. Finally, execute the Java agent's diagnostic test to verify that the agent is installed correctly and can communicate with Contrast using the configuration parameters from `/etc/contrast/java/contrast_security.yaml`:

```
$ java -jar /opt/contrast/contrast-agent.jar diagnostic
*** Contrast Agent (version 3.6.3-SNAPSHOT)
[!] Attempting to connect to the Contrast TeamServer at https://
apptwo.contrastsecurity.com/Contrast (No proxy).
[!] Attempting to resolve domain: apptwo.contrastsecurity.com
    Resolved domain apptwo.contrastsecurity.com to IP Address
52.200.215.12
[+] Client successfully resolved the DNS of the Contrast TeamServer. No
proxy needed.
```

```
[!] Issuing HTTP request to Contrast...
    Executing request...
    Reading response [200]
    Response size = 4209
    Snippet: <!doctype html> <!--[if gt IE 8]><!--> <html class="no-
js" i
[+] Client can connect directly to the Contrast TeamServer. No proxy
needed.
```

Scala

You can use the Contrast Java agent with Contrast Assess or Contrast SCA to analyze Scala-based applications.

The Java agent analyzes Scala web applications built on traditional application servers, and newer Scala web applications such as those built with Play. If there's a JVM, the Scala agent can provide security insights.

As your application runs, the Java agent's sensors gather information about the application's security, architecture and libraries. You can see the results of the agent's analysis in Contrast.

The Scala agent supports these Contrast features:

- Route coverage
- Flow maps
- SCA library discovery

Kotlin

You can use the Contrast Java agent with Contrast Assess or Contrast SCA to analyze Kotlin-based applications.

The Java agent analyzes Kotlin web applications built on traditional application servers, and newer Kotlin server-side applications, such as SpringBoot.


If there's a JVM, the Kotlin agent can provide security insights. As your application runs, the Java agent's sensors gather information about the application's security, architecture and libraries. You can see the results of the agent's analysis in Contrast.

Run your application as you would with the Contrast Java agent. Kotlin support is automatic.

Java application servers



NOTE

This documentation provides content for supported content. Links indicated by the  icon take you to other documentation that may be helpful.

The following application servers are available:

-  [Axis2](#)
-  [Glassfish](#)
- [JBoss / Wildfly \(page 143\)](#)
- [Jetty \(page 144\)](#)

- [Tomcat \(page 145\)](#)
- [Weblogic \(page 146\)](#)
- [WebSphere \(page 147\)](#)

See also

- [Install the Java agent \(page 123\)](#)
- [Supported technologies \(page 120\)](#)
- [Configure the Java agent \(page 149\)](#)

Configure the Java agent for JBoss EAP, JBoss AS or WildFly



CAUTION

Be careful not to confuse version numbers. JBoss EAP prior to version 7 is based on JBoss AS. JBoss EAP 7.X is based on WildFly.

Run JBoss with the Java agent

1. Download the Java agent JAR from one of these repositories:
 - [Maven Central \(page 124\)](#)
 - [Debian \(page 125\)](#)
 - [RPM \(page 126\)](#)
2. You can either run JBoss from a BAT file, or in domain mode.
 - **BAT file:** If you run JBoss from **Windows** using: *domain.bat*, *standalone.bat*, *standalone.conf.bat*, or *run.bat* or **Unix** using: *domain.sh*, *domain.conf*, *standalone.sh*, or *standalone.conf* with a *.conf.bat* file, modify the configuration file. It should enable the Contrast JVM parameters and return to the start-up script.

To do this, replace `<YourContrastJarPath>` with the path to your [Contrast JAR \(page 120\)](#) file, and use the JBoss server directory for your environment. Then add this line to the end of your *.conf.bat* file:

- **Windows:**

```
set "JAVA_OPTS=-javaagent:<YourContrastJarPath> %JAVA_OPTS%"
```

- **Unix:**

```
JAVA_OPTS="-javaagent:<YourContrastJarPath> $JAVA_OPTS"
```

- **Domain mode:** If you run JBoss 6 EAP or JBoss AS 7.X in Domain mode using *domain.bat* or *domain.sh*, you must add the `-javaagent` switch to the JVM options in `$JBOSS_HOME/domain/configuration/domain.xml`.

In this example, replace `<YourContrastJarPath>` with the path to your [Contrast JAR \(page 120\)](#) file:

```
<server-group ...>
  <jvm name="default">
    <jvm-options>
      <option value="-javaagent:<YourContrastJarPath>" />
    </jvm-options>
  </jvm>
  ...
</server-group>
```

Use WildFly with Java 2 security manager

You can configure the Java agent when using WildFly with [Java 2 security \(page 210\)](#). WildFly versions 9 through 20 are supported. WildFly 8 is not supported.

To enable the Java 2 security manager in Wildfly:

1. Either pass a command-line argument `-secmgr`, or set an environment variable `SECMGR` to true:

```
SECMGR="true"
```

2. To enable permissions for the Java agent, append this Contrast policy to `$JAVA_HOME/jre/lib/security/java.policy` (for JDK 6-8), or `$JAVA_HOME/lib/security/default.policy` (for JDK 9 and later). Replace `<YourContrastJarPath>` with the path to your [Contrast JAR \(page 120\)](#) and use:

```
grant codeBase "file:<YourContrastJarPath>" {
    permission java.security.AllPermission;
};
```

3. To allow the agent to function with Wildfly's classloader system, modify the value of the environment variable `JBoss_MODULES_SYSTEM_PKGS` (originally `org.jboss.byteman`), to also include the Java agent base package: `com.contrastsecurity.agent,org.jboss.byteman`



TIP

Learn more about using [Java EE 7 security manager with WildFly](#), or read the [default policy implementation and policy file syntax](#).

Configure the Java agent for Jetty

To configure the Java agent with a Jetty distribution:

1. Download the Java agent JAR from one of these repositories:
 - [Maven Central \(page 124\)](#)
 - [Debian \(page 125\)](#)
 - [RPM \(page 126\)](#)
2. On your Jetty environment, replace `<YourContrastJarPath>` with the path to your [Contrast JAR \(page 120\)](#) file. Then add the following line to your `<JettyDirectory>/start.ini` file:

```
-javaagent:<YourContrastJarPath>
```

3. If you are using Java 2 security manager, create a `contrast.policy` file that contains this code. (Replace `<YourContrastJarPath>` with the path to your [Contrast JAR \(page 120\)](#) file.)

```
grant codeBase "file:<YourContrastJarPath>" {
    permission java.security.AllPermission;
};
```

Then complete these configurations:

- **Jetty 7-8:** Copy the file to the `JETTY_HOME/lib/policy` folder. Add `--secure` to the `JETTY_ARGS` environment variable.
- **Jetty 9:** Add the policy you created to your own configured policy. Replace `<YourPolicy>` with the name of your policy and enable the security manager with the standard environment variable settings:

```
-Djava.security.manager -Djava.security.policy=<YourPolicy>
```

**IMPORTANT**

Jetty 9 and later do not officially support security management policies.

**TIP**

See the [Jetty Policy](#) for more information about using Jetty with Java 2 security manager.

Configure the Java agent for Tomcat

First, download the Java agent JAR from one of these repositories:

- [Maven Central \(page 124\)](#)
- [Debian \(page 125\)](#)
- [RPM \(page 126\)](#)

Use the guidelines below to configure the Java agent depending on how you run Contrast with Tomcat.

Run from Windows or Unix

The `CATALINA_OPTS` environment variable is used to pass configuration flags and system properties to the JVM that runs the Tomcat server.

Tomcat recommends using a `setenv` script to specify environment variables. You can learn more about the `setenv` script, including how to find or create it as needed, by consulting `RUNNING.txt`, which is included with every distribution of Tomcat.

To enable Contrast add the `-javaagent` configuration to `CATALINA_OPTS` in either `setenv.sh`, if running on a Unix-like operating system, or `setenv.bat` if running on Windows. For example:

- **Windows:**

```
set "CATALINA_OPTS=%CATALINA_OPTS% -javaagent:<YourContrastJarPath>"
```

- **Unix:**

```
export CATALINA_OPTS="$CATALINA_OPTS -javaagent:<YourContrastJarPath>"
```

Run on the Tomcat service in Windows

1. If you run Tomcat as a service, open the Tomcat service manager and change the JVM options to add the agent.
2. Double-click the Tomcat icon in the system tray (or right-click and select **Configure**). (If the icon isn't there, you might have to start it manually by running `tomcat9w.exe` in the Tomcat bin directory.)
3. Switch to the **Java** tab to see where you need to add the `-javaagent` flag.

Run Tomcat with Java 2 security

1. Create a `contrast.policy` file that contains this code (or append it to the `catalina.policy` file). Replace `<YourContrastJarPath>` with the path to your [Contrast JAR \(page 120\)](#) file. For example:

```
grant codeBase "file:<YourContrastJarPath>" {  
    permission java.security.AllPermission;  
};
```

2. Append the *contrast.policy* file to the *\$CATALINA_HOME/conf/catalina.policy* file. No additional configuration is needed. Run your Tomcat installation with command-line parameter *-security*.

Configure the Java agent for WebLogic

First, download the Java agent JAR from one of these repositories:

- [Maven Central \(page 124\)](#)
- [Debian \(page 125\)](#)
- [RPM \(page 126\)](#)

Use the guidelines below to configure the Java agent depending on how you run Contrast with WebLogic.

Unix

1. If you launch WebLogic yourself, you must add Contrast's JVM parameter to the *startWebLogic* file in your installation's *bin* directory. For UNIX-based operating systems, the path to this file looks like:

```
/path/to/appserver/userprojects/domains/base_domain/bin/startWebLogic.sh
```

2. In this file, add the Contrast engine as a *-javaagent* to the *JAVA_OPTIONS* environment variable before the Java execution step. Replace *<YourContrastJarPath>* with the path to your [Contrast JAR \(page 120\)](#) file. For example:

```
export JAVA_OPTIONS="$JAVA_OPTIONS -javaagent:<YourContrastJarPath>"
```

Windows

1. For Windows systems, the path looks like:

```
C:\Oracle\Middleware\userprojects\domains\base_domain\bin\startWebLogic.bat
```

2. At the beginning of the file, add the Contrast engine as a *-javaagent* to the *JAVA_OPTIONS* environment variable. Replace *<YourContrastJarPath>* with the path to your [Contrast JAR \(page 120\)](#) file. Substitute the WebLogic server for your environment. For example:

```
set "JAVA_OPTIONS=%JAVA_OPTIONS% -javaagent:<YourContrastJarPath>"
```

Use Java 2 with WebLogic

1. Create a *contrast.policy* file that contains this code (or append it to the *weblogic.policy* file). Replace *<YourContrastJarPath>* with the path to your [Contrast JAR \(page 120\)](#) file. For example:

```
grant codeBase "file:<YourContrastJarPath>" {  
    permission java.security.AllPermission;  
};
```

2. WebLogic includes a template file under *@WL_HOME/server/lib/weblogic.policy* which contains the suggested starting point for booting a WebLogic server with the security manager enabled. Older versions of WebLogic (10 and prior) will require *@WL_HOME* in the template file replaced with the actual path to the root directory of the WebLogic install.
3. When the security manager is enabled, the policy *@WL_HOME/server/lib/weblogic.policy* file acts as the default. Otherwise, a custom policy file may be specified with *-Djava.security.policy==<YourPath>* where *<YourPath>* is the path to your custom file. The *==* is important as it overrides the default path setting that WebLogic boots with.

**TIP**

For more information read about [using Java Security to protect WebLogic resources](#).

Configure the Java agent for WebSphere

First, download the Java agent JAR from one of these repositories:

- [Maven Central \(page 124\)](#)
- [Debian \(page 125\)](#)
- [RPM \(page 126\)](#)

Use the guidelines below to configure the Java agent, depending on how you run Contrast with WebSphere.

**NOTE**

IBM J9 doesn't allow the Java Instrumentation API to alter core Java classes when using the [Shared Classes](#) feature. You must disable this feature by specifying `-Xshareclasses:none` in your JVM parameters, as shown above.

Similarly, if `-Dcom.ibm.oti.shared.enabled=true` is set, you may also run into problems in older J9 JREs.

WebSphere trust and key store

WebSphere maintains its own trust and key store, separate from the trust store included as part of the Java JRE. The agent starts before WebSphere is initialized and so the WebSphere specific trust store is not configured. Therefore, the agent uses the default trust store located in the Java `JRE/lib/security/cacerts` file, unless extra configuration is provided to the JVM.

However, in some scenarios, (like requiring a proxy server that uses internal only or self-signed certificates) specific extra steps are necessary. You can either:

1. Install the required certs into both the JRE `cacerts` trust store and also the WebSphere specific trust store. This means the certificate chain can be validated by both the agent and also your web application.
2. Provide Java with the standard trust store system properties to change the trust store to be the same as the WebSphere trust store. This has the advantage of only requiring the certificate to be installed in one location: the WebSphere trust store. For example:

```
-Djavax.net.ssl.trustStore=opt/IBM/WebSphere/AppServer/profiles/AppSrv01/  
config/cells/DefaultCell01/nodes/DefaultNode01/trust.p12  
-Djavax.net.ssl.trustStoreType=PKCS12  
-Djavax.net.ssl.trustStorePassword=secret
```

WebSphere itself supports methods of encoding the password but these are not available when setting the trust store password for the agent, as it is executing before WebSphere starts.

Add Contrast with WebSphere

If you launch WebSphere yourself, add Contrast's JVM parameter to the `server.xml` file in your cell directory. Replace `<CellName>` and `<NodeName>` with the name of the cell and node. Replace `<YourContrastJarPath>` with the path to your [Contrast JAR \(page 120\)](#) file. For example:

```
<WebSphereDirectory>\AppServer\profiles\AppSrv01\config\cells\<CellName>\nodes\<NodeName>\servers\server1\server.xml

<jvmEntries genericJvmArguments="-javaagent:<YourContrastJarPath>
-Xshareclasses:none">
    ...
</jvmEntries>
```

Add Contrast with the WebSphere Administration Console

You can also add Contrast through the WebSphere administration console by following instructions from the [WebSphere support site](#).

Use Java 2 with WebSphere

1. Create a *contrast.policy* file that contains this code (or append it to the *server.policy* file). Replace *<YourContrastJarPath>* with the path to your [Contrast JAR \(page 120\)](#) file. For example:

```
grant codeBase "file:<YourContrastJarPath>" {
    permission java.security.AllPermission;
};
```

2. Append the *contrast.policy* file to the *\$WEBSPPHERE_HOME/AppServer/profiles/AppSrv01/properties/server.policy*.
3. Enable the security manager with the wsadmin tool:
 - **Jacl:** `$AdminTask setAdminActiveSecuritySettings {-enforceJava2Security true}`
 - **Jython:** `AdminTask.setAdminActiveSecuritySettings(' -enforceJava2Security true')`



TIP

Learn more about [Java security manager](#) and [enabling and disabling Java 2 security manager using scripting](#).

Java Quick Start Guide

Contrast uses agents to install sensors that monitor your code for vulnerabilities. Agents analyze for vulnerabilities in development environments and look for attacks in runtime production environments.

As your application runs, the agent analyzes information (such as HTTP requests, data flow, backend connections, and library dependencies) and sends vulnerabilities and attacks to Contrast where you can view, prioritize, and take immediate action on them.

This guide should get Contrast up and running on your application in just a few minutes, so you can see how it works.



TIP

For future installations, you may want to consider your organization's build tools and deployment pipeline, your security goals and the environments where you want to use Contrast. You can read about [other methods to install Contrast \(page 1051\)](#) that may better adapt to your situation.

Prerequisites

This guide assumes you use an application that meets these prerequisites:

- The application must have access to the internet without using a proxy.
- Your web application is packaged in a JAR file.
- It must use [supported versions, frameworks, and tools \(page 120\)](#).

You will also need access to a command line interface (with a chosen directory for downloading the agent) and your organization's instance of Contrast.

Install

1. Start the Java agent wizard:
 - a. In the Contrast web interface, select **Add new**.
 - b. Select the **Application** card.
 - c. Select **Java** as the language.
 - d. Select an operating system.
2. Under Select application deployment method, select **Install manually**.
3. Copy the displayed command to download the agent from [Maven Central \(page 124\)](#).
4. To configure the agent, under Configure the agent, select **Use Connection Token** and copy the displayed commands to set this environment variable:

```
CONTRAST__API__TOKEN
```

Agent Token: This variable is a base64 encoded JSON object containing the url, api_key, service_key, and user_name configuration settings, allowing you to set them set in a single variable.

If your agent configuration refers to both the legacy settings and the agent token, (in environment variables or the YAML file), the legacy settings take precedence. To use just the agent token value, make sure you remove references to the legacy settings.

Legacy settings: If you are using a Java agent version earlier than 6.10.1, under Configure the agent, select **Use API configuration**, and copy the displayed commands to set these environment variables:

```
CONTRAST__API__URL
CONTRAST__API__API_KEY
CONTRAST__API__SERVICE_KEY
CONTRAST__API__USER_NAME
```

5. Under Configure application server, select an application server,
6. Copy the displayed commands to complete the configuration.
7. To verify that Contrast is working, use your application as you normally would. For example, click on your application's web interface, or send some API commands.
Then in the Contrast web interface, select **Applications** in the header. You should see the name of your application.
You can also select **Server** in the header and you should see the hostname of your (local) server listed here.

Configure the Java agent

The [standard installation \(page 102\)](#) for all agents uses this [order of precedence \(page 106\)](#).

You can configure the Java agent using:

- Java system properties
- [Environment variables \(page 110\)](#)
- Java YAML template

**TIP**

Use the [Contrast agent configuration editor \(page 109\)](#) to create or upload a YAML configuration file, validate YAML and get setting recommendations.

You may need to configure your application's Java environment to work effectively with the agent if your system uses:

- [Multi-tenant application configuration](#): If your JVM application server hosts multiple applications during a deployment, you can distinguish applications from each other and then apply individual configuration options.
[Multi-tenant application configuration](#): If your JVM application server hosts multiple applications during a deployment, you can distinguish applications from each other and then apply individual configuration options.
- [TLS certificates \(page 209\)](#)
- [Java 9 Modules \(page 210\)](#)
- [Java 2 Security \(page 210\)](#)
- [Integrations: \(page 1051\)](#) The Contrast Java agent, can also be configured and run in conjunction with several third-party tools, plugins and integrations. Consult the remote product documentation for information about how other products work.

Java system properties

Substitute `<YourContrastJarPath>` with the path to your [Contrast JAR \(page 120\)](#), and use these commands to learn more about system properties:

- To generate a list of general properties using the Contrast agent JAR, use:

```
java -jar <YourContrastJarPath> properties
```

- Use command line with tools to search for commands. For example, these commands display a list of proxy-related properties:

Using the built-in filter:

```
java -jar <YourContrastJarPath> properties --filter=proxy
```

Java YAML configuration template

Use this template to configure the Java agent using a YAML configuration file. (Learn more about [YAML configuration \(page 107\)](#).)

Place your YAML file in the default location:

- **Unix:** `/etc/contrast/java/contrast_security.yaml`
- **Windows:** `C:/ProgramData/contrast/java/contrast_security.yaml`

```
#
=====
==
# Use the properties in this YAML file to configure a Contrast agent.
# Go to https://docs.contrastsecurity.com/en/order-of-precedence.html
```

```
# to determine the order of precedence for configuration values.
#
=====
==

# Use this setting if you want to temporarily disable a Contrast agent.
# Set to `true` to enable the agent; set to `false` to disable the agent.
# enable: true

#
=====
==
# api
# Use the properties in this section to connect the agent to the Contrast
UI.
#
=====
==
api:

# ***** REQUIRED *****
# Set the URL for the Contrast UI.
url: https://app.contrastsecurity.com/Contrast

# ***** REQUIRED *****
# Set the API key needed to communicate with the Contrast UI.
api_key: NEEDS_TO_BE_SET

# ***** REQUIRED *****
# Set the service key needed to communicate with the Contrast
# UI. It is used to calculate the Authorization header.
service_key: NEEDS_TO_BE_SET

# ***** REQUIRED *****
# Set the user name used to communicate with the Contrast
# UI. It is used to calculate the Authorization header.
user_name: NEEDS_TO_BE_SET

# base64 encoded JSON object containing the `url`,
# `api_key`, `service_key`, and `user_name` config options,
# allowing them all to be set in a single variable.
# token: NEEDS_TO_BE_SET

# Set the timeout for communicating with TeamServer. This property will be
# respected over the deprecated legacy configuration *contrast.timeout*.
# timeout_ms: NEEDS_TO_BE_SET

#
=====
# api.proxy
# Use the following properties for communication
# with the Contrast UI over a proxy.
#
=====
```

```
# proxy:

# Set value to `true` for the agent to communicate with
# the Contrast web interface over a proxy. Set value to
# `false` if you don't want to use the proxy. If no value is
# indicated, the presence of a valid **contrast.proxy.host**
# and **contrast.proxy.port** will enable the proxy.
# enable: NEEDS_TO_BE_SET

# Set the proxy host. It must be set with port and scheme.
# host: localhost

# Set the proxy port. It must be set with host and scheme.
# port: 1234

# Set the proxy scheme (e.g., `http` or
# `https`). It must be set with host and port.
# scheme: http

# Set this property as an alternate for `scheme://host:port`. It takes
# precedence over the other settings, if specified; however, an error
# will be thrown if both the URL and individual properties are set.
# url: NEEDS_TO_BE_SET

# Set the proxy user.
# user: NEEDS_TO_BE_SET

# Set the proxy password.
# pass: NEEDS_TO_BE_SET

# Set the proxy authentication type. Value
# options are `NTLM`, `Digest`, and `Basic`.
# auth_type: NEEDS_TO_BE_SET

#
=====
==
# agent
# Use the properties in this section to control the way and frequency
# with which the agent communicates to logs and the Contrast UI.
#
=====
==
# agent:

#
=====
# agent.diagnostics
# Use the properties in this section to specify the information the agent
# should collect and report in order to diagnose problems in the agent.
#
#
=====
# diagnostics:
```

```
# Creates config and system info files
# at startup if true. True by default.
#
# The same thing can be achieved by setting the
# CONTRAST_AGENT_DIAGNOSTICS_ENABLE=[true/false] env variable.
#
# enable: true

#
=====
# agent.route_coverage
# Use the following properties for the route-based coverage feature.
#
=====
# route_coverage: {}

#
=====
# agent.reporting
# Use the following settings to configure reporting to the Contrast UI.
#
=====
# reporting:

# Set the grace period (in milliseconds) after
# agent shutdown to allow draining pending reports.
# shutdown_grace_period_ms: 120000

#
=====
# agent.effective_config
# None
#
=====
# effective_config:

#
=====
# agent.effective_config.reporting
# None
#
=====
# reporting:

# Defaults to `true`. Controls whether configuration
# setting reports are sent to the Contrast web interface.
# enable: true

#
=====
# agent.logger
# Define the following properties to set logging values.
# If the following properties are not defined, the
# agent uses the logging values from the Contrast UI.
#
```

```
=====
# logger:

# Enable diagnostic logging by setting a path to a log file.
# While diagnostic logging hurts performance, it generates
# useful information for debugging Contrast. The value set here
# is the location to which the agent saves log output. If no
# log file exists at this location, the agent creates a file.
#
# Example - `/opt/Contrast/contrast.log` creates a log in the
# `/opt/Contrast` directory, and rotates it automatically as needed.
#
# path: ./contrast_agent.log

# Set the the log output level. Valid options are
# `ERROR`, `WARN`, `INFO`, `DEBUG`, and `TRACE`.
# level: INFO

# Set to `true` to redirect all logs to
# `stdout` instead of the file system.
# stdout: false

# Set to `true` to redirect all logs to `stderr` instead of
# the file system. May be combined with the corresponding
# `stdout` configuration to write to both streams.
# stderr: false

# Change the Contrast logger from a file-sized based rolling scheme
# to a date-based rolling scheme. At midnight server time, the
# previous day log is renamed to *file_name.yyyy-MM-dd*. Note -
# this scheme does not have a size limit; manual log pruning is
# required. You must set this flag to use the backups and size flags.
# roll_daily: false

# Set the roll size for log files in megabytes. The agent will
# attempt to prevent the log file from being larger than this size.
# roll_size: 100

# Set the number of backup files to keep. Set to `0` to disable.
# backups: 10

#
=====
# agent.security_logger
# Define the following properties to set security logging
# values associated with Protect. If not defined, the agent
# uses the security logging (CEF) values from the Contrast UI.
#
=====
# security_logger:

# Set the file to which the agent logs security events.
# path: ./contrast/security.log

# Set the log level for security logging. Valid options
```

```
# are `ERROR`, `WARN`, `INFO`, `DEBUG`, and `TRACE`.
# level: ERROR

# Change the Contrast security logger from a file-sized based rolling
# scheme to a date-based rolling scheme. At midnight server time,
# the log from the previous day is renamed to *file_name.yyyy-MM-dd*.
# Note - this scheme does not have a size limit; manual log
# pruning will be required. This flag must be set to use the
# backups and size flags. Value options are `true` or `false`.
# roll_daily: NEEDS_TO_BE_SET

# Specify the file size cap (in MB) of each log file.
# roll_size: NEEDS_TO_BE_SET

# Specify the number of backup logs that the agent will create before
# Contrast cleans up the oldest file. A value of `0` means that no
backups
# are created, and the log is truncated when it reaches its size cap.
#
# Note - this property must be used with
# `agent.security_logger.roll_daily=false`; otherwise,
# Contrast continues to log daily and disregard this limit.
#
# backups: NEEDS_TO_BE_SET

#
=====
# agent.security_logger.syslog
# Define the following properties to set Syslog values. If the
properties
# are not defined, the agent uses the Syslog values from the Contrast
UI.
#
=====
# syslog:

# Set to `true` to enable Syslog logging.
# enable: NEEDS_TO_BE_SET

# Set the IP address of the Syslog server
# to which the agent should send messages.
# ip: NEEDS_TO_BE_SET

# Set the port of the Syslog server to
# which the agent should send messages.
# port: NEEDS_TO_BE_SET

# Set the facility code of the messages the agent sends to Syslog.
# facility: 19

# Set the log level of Exploited attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_exploited: ALERT

# Set the log level of Blocked attacks. Value options are `ALERT`,
```

```
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_blocked: NOTICE

# Set the log level of Blocked At Perimeter
# attacks. Value options are `ALERT`, `CRITICAL`,
# `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_blocked_perimeter: NOTICE

# Set the log level of Probed attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_probed: WARNING

# Set the log level of Suspicious attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_suspicious: WARNING

#
=====
# agent.security_logger.syslog.heartbeat
# Define the following properties to
# set the Syslog heartbeat properties.
#
=====
# heartbeat:

# Set to `true` to enable the Syslog heartbeat.
# The heartbeat will issue a Syslog message at
# the INFO level after every interval passes.
# enable: false

# Set the interval for sending heartbeat messages
# to the Syslog server (in milliseconds).
# interval_ms: 60000

#
=====
# agent.java
# The following properties apply to any Java agent-wide configurations.
#
=====
# java:

# Configure the Java agent to skip its application discovery
# algorithm, and instead associate all libraries, vulnerabilities,
# and web traffic to a single application with the name specified
# by this property. This configuration is preferred when deploying
# Java SE applications with embedded web servers (e.g., applications
# built with Spring Boot, Dropwizard, and embedded Jetty). When used
# with an application server, this configuration associates all
# web traffic with the single, standalone application, including
# web traffic handled by application server-hosted endpoints that
# would not be associated with a discovered application otherwise.
#
# Note - This settings takes preferences
# over the `application.name` setting.
```



```
#
# standalone_app_name: NEEDS_TO_BE_SET

# By default, the Java agent visits all classes at startup to look
# for vulnerabilities, which the agent may detect by scanning a
# class (e.g., hardcoded passwords). Set this property to `false`
# to disable the default behavior. If disabled, the agent will
# only visit classes which are likely to require sensors; this
# can improve application startup time, but may produce fewer
# findings (most likely findings that require static analysis).
#
# scan_all_classes: true

# By default, the Java agent deeply inspects all JAR and WAR files
loaded
# by the JVM to build a comprehensive understanding of the type
hierarchy.
# This understanding allows Contrast to instrument sensors into types
# that it might have overlooked. In most cases, this produces a slight
# increase in accuracy at the cost of increased application startup
# time. Set this property to `false` to disable this level of
inspection.
#
# scan_all_code_sources: true

#
=====
==
# inventory
# Use the properties in this section to override the inventory features.
#
=====
==
# inventory:

# Set to `false` to disable inventory features in the agent.
# enable: true

# Define a list of directories where libraries are stored.
# Directories must be formatted as a semicolon-delimited
# list for Windows or a colon-delimited list for Linux.
#
# Define a list of directories where libraries are stored.
# Directories must be formatted as a semicolon-delimited list.
# Example - `path1;path2;path3` (Windows) or `path1:path2:path3` (Linux)
# Example - `path1:path2:path3`
#
# library_dirs: NEEDS_TO_BE_SET

# Set the maximum archive unpacking depth when analyzing libraries.
# library_depth: 10

# Set the boolean to more aggressively limit the
# manifest information reported for libraries. If true,
# the limit is 1,000 characters, otherwise it's 3,000.
```

```
# prune_package_details: true

# Apply a list of labels to libraries. Labels
# must be formatted as a comma-delimited list.
# Example - `label1, label2, label3`
#
# tags: NEEDS_TO_BE_SET

#
=====
==
# assess
# Use the properties in this section to control Assess.
#
=====
==
# assess:

# Include this property to determine if the Assess
# feature should be enabled. If this property is not
# present, the decision is delegated to the Contrast UI.
# enable: false

# Control the values captured by Assess vulnerability events. `Full`
# captures most values by calling ToString on objects, which can
# provide more info but causes increased memory usage. `Minimal`
# has better performance as it only captures String type objects
# as strings and uses type name for other object type values.
# event_detail: minimal

# Apply a list of labels to vulnerabilities and preflight
# messages. Labels must be formatted as a comma-delimited list.
# Example - `label1, label2, label3`
#
# tags: NEEDS_TO_BE_SET

#
=====
# assess.sampling
# Use the following properties to control sampling in the agent.
#
=====
# sampling:

# Set to `true` to enable sampling.
# enable: false

# This property indicates the number of requests
# to analyze in each window before sampling begins.
# baseline: 5

# This property indicates that every *nth*
# request after the baseline is analyzed.
# request_frequency: 10
```

```
# This property indicates the duration for which a sample set is valid.
# window_ms: 180_000

#
=====
# assess.rules
# Use the following properties to control simple rule configurations.
#
=====
# rules:

# Define a list of Assess rules to disable in the agent. To view a
# list of rule names, in Contrast go to user menu > Policy Management >
# Assess rules. The rules must be formatted as a comma-delimited list.
#
# Example - Set `reflected-xss,sql-injection` to disable
# the reflected-xss rule and the sql-injection rule.
#
# disabled_rules: NEEDS_TO_BE_SET

#
=====
==
# profile
# Set configuration values under a profile name to enable
# multi-tenant application configuration on web servers. See
# https://support.contrastsecurity.com/hc/en-us/articles/360052187171-Multi-Application-configuration-with-Contrast-Profiles
# for more details.
#
=====
==
# profile: {}

#
=====
==
# protect
# Use the properties in this section to override Protect features.
#
=====
==
# protect:

# Include this property to determine if the Protect
# feature should be enabled. If this property is not
# present, the decision is delegated to the Contrast UI.
# enable: false

#
=====
# protect.rules
# Use the following properties to set simple rule configurations.
#
=====
```

```
# rules:

# Define a list of Protect rules to disable in the agent. To view a
# list of rule names, in Contrast go to user menu > Policy Management >
# Protect rules. The rules must be formatted as a comma-delimited list.
# disabled_rules: NEEDS_TO_BE_SET

#
=====
# protect.rules.bot-blocker
# Use the following selection to configure if the
# agent blocks bots. Set to `true` to enable blocking.
#
=====
# bot-blocker:

# Set to `true` for the agent to block known bots.
# enable: false

#
=====
# protect.rules.sql-injection
# Use the following settings to configure the sql-injection rule.
#
=====
# sql-injection:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or off.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

# Tell the agent to detect when semantic analysis of the query
# reveals tautologies used in exfiltration attacks (e.g., "or
# 1=1" or "or 2<>3"). The agent blocks if blocking is enabled.
# detect_tautologies: false

# Tell the agent to detect when semantic analysis of the query
# reveals the invocation of dangerous functions typically used in
# weaponized exploits. The agent blocks if blocking is enabled.
# detect_dangerous_functions: false

# Tell the agent to detect when semantic analysis of the query
# reveals chained queries, which is uncommon in normal usage but
# common in exploit. The agent blocks if blocking is enabled.
# detect_chained_queries: false

# Tell the agent to detect when semantic analysis of the query
# reveals database queries are being made for system tables and
# sensitive information. The agent blocks if blocking is enabled.
# detect_suspicious_unions: false
```

```
# Tell the agent to be more aggressive in detecting user
# inputs as SQL comments. This enables the agent to better
# detect SQL Injection input vectors that use comments to
# terminate queries. The agent blocks if blocking is enabled.
# aggressive_comment: false

#
=====
# protect.rules.cmd-injection
# Use the following properties to configure
# how the command injection rule works.
#
=====
# cmd-injection:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

# Detect when the agent sees user parameters being executed as
# system commands. The agent blocks if blocking is enabled.
# detect_parameter_command_backdoors: true

# Detect when a system command is issued which contains
# chained commands. The agent blocks if blocking is enabled.
# detect_chained_commands: true

# Detect when a system command is issued with an argument matching a
# known dangerous file path. The agent blocks if blocking is enabled.
# detect_dangerous_path_args: true

# Tell the agent to detect when commands come directly
# from input. The agent blocks if blocking is enabled.
# detect_phased_commands: true

#
=====
# protect.rules.cmd-injection-process-hardening
# Use the following settings to configure whether
# the agent blocks all attempts to start an external
# process. To enable blocking, set to 'true'.
#
=====
# cmd-injection-process-hardening:

# Set to `true` to enable the agent to block
# all attempts to start external processes.
# enable: false

#
=====
```

```
# protect.rules.path-traversal
# Use the following properties to configure
# how the path traversal rule works.
#
=====
# path-traversal:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

# Detect when custom code attempts to access sensitive
# system files. The agent blocks if blocking is enabled.
# detect_custom_code_accessing_system_files: true

# Detect when users attempt to bypass filters by
# using "::$DATA" channels or null bytes in file
# names. The agent blocks if blocking is enabled.
# detect_common_file_exploits: true

#
=====
# protect.rules.method-tampering
# Use the following properties to configure
# how the method tampering rule works.
#
=====
# method-tampering:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.reflected-xss
# Use the following properties to configure how
# the reflected cross-site scripting rule works.
#
=====
# reflected-xss:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
```

```
#
# mode: off

#
=====
# protect.rules.xxe
# Use the following properties to configure
# how the XML external entity works.
#
=====
# xxe:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.padding-oracle
# Use the following properties to configure
# how the padding-oracle rule works.
#
=====
# padding-oracle: {}

#
=====
==
# application
# Use the properties in this section for
# the application(s) hosting this agent.
#
=====
==
# application:

# Override the reported application name.
#
# Note - On systems where multiple, distinct applications may be served
# by a single process, this configuration causes the agent to report
# all discovered applications as one application with the given name.
#
# Note - On Java systems where multiple, distinct applications may be
# served by a single process, this configuration causes the agent to
report
# all discovered applications as one application with the given name.
#
# name: NEEDS_TO_BE_SET

# Override the reported application path.
# path: NEEDS_TO_BE_SET
```

```
# Add the name of the application group with which this
# application should be associated in the Contrast UI.
# group: NEEDS_TO_BE_SET

# Add the application code this application should use in the Contrast UI.
# code: NEEDS_TO_BE_SET

# Override the reported application version.
# version: NEEDS_TO_BE_SET

# Apply labels to an application. Labels must
# be formatted as a comma-delimited list.
# Example - `label1,label2,label3`
#
# tags: NEEDS_TO_BE_SET

# Define a set of `key=value` pairs (which conforms to RFC 2253) for
# specifying user-defined metadata associated with the application. The
# set must be formatted as a comma-delimited list of `key=value` pairs.
# Example - `business-unit=accounting, office=Baltimore`
#
# metadata: NEEDS_TO_BE_SET

# Provide the ID of a session which already exists in the Contrast
# UI. Vulnerabilities discovered by the agent are associated with
# this session. If an invalid ID is supplied, the agent will be
# disabled. This option and `application.session_metadata` are
# mutually exclusive; if both are set, the agent will be disabled.
# session_id: NEEDS_TO_BE_SET

# Provide metadata which is used to create a new session ID in the
# Contrast UI. Vulnerabilities discovered by the agent are associated with
# this new session. This value should be formatted as `key=value` pairs
# (conforming to RFC 2253). Available key names for this configuration
# are branchName, buildNumber, commitHash, committer, gitTag, repository,
# testRun, and version. This option and `application.session_id` are
# mutually exclusive; if both are set the agent will be disabled.
# session_metadata: NEEDS_TO_BE_SET

#
=====
==
# server
# Use the settings in this section to set metadata for the server
# hosting this agent. Contrast recognizes common, supported server
# names, paths, types and environments. Doing this may require a new
# server or license, and it may affect functionality of some features.
#
=====
==
# server:

# Override the reported server name.
# name: localhost
```



```
# Override the reported server path.
# path: NEEDS_TO_BE_SET

# Override the reported server type.
# type: NEEDS_TO_BE_SET

# Set the environment directly to override the default set
# by the Contrast UI. This allows the user to configure the
# environment dynamically at startup rather than manually
# updating the Server in the Contrast UI themselves afterwards.
#
# Valid values include `QA`, `PRODUCTION` and `DEVELOPMENT`.
# For example, `PRODUCTION` registers this Server as
# running in a `PRODUCTION` environment, regardless of the
# organization's default environment in the Contrast UI.
#
# environment: NEEDS_TO_BE_SET

# Apply a list of labels to the server. Labels
# must be formatted as a comma-delimited list.
# Example - `label1,label2,label3`
#
# tags: NEEDS_TO_BE_SET

# Set to `false` to disable detection of cloud
# provider metadata such as resource identifiers.
# discover_cloud_resource: true
```

```
#
=====
==
# Use the properties in this YAML file to configure a Contrast agent.
# Go to https://docs.contrastsecurity.com/en/order-of-precedence.html
# to determine the order of precedence for configuration values.
#
=====
==

# Use this setting if you want to temporarily disable a Contrast agent.
# Set to `true` to enable the agent; set to `false` to disable the agent.
# enable: true

#
=====
==
# api
# Use the properties in this section to connect the agent to the Contrast
# UI.
#
=====
==
api:
```

```
# ***** REQUIRED *****
# Set the URL for the Contrast UI.
url: https://app.contrastsecurity.com/Contrast

# ***** REQUIRED *****
# Set the API key needed to communicate with the Contrast UI.
api_key: NEEDS_TO_BE_SET

# ***** REQUIRED *****
# Set the service key needed to communicate with the Contrast
# UI. It is used to calculate the Authorization header.
service_key: NEEDS_TO_BE_SET

# ***** REQUIRED *****
# Set the user name used to communicate with the Contrast
# UI. It is used to calculate the Authorization header.
user_name: NEEDS_TO_BE_SET

# base64 encoded JSON object containing the `url`,
# `api_key`, `service_key`, and `user_name` config options,
# allowing them all to be set in a single variable.
# token: NEEDS_TO_BE_SET

# Set the timeout for communicating with TeamServer. This property will be
# respected over the deprecated legacy configuration *contrast.timeout*.
# timeout_ms: NEEDS_TO_BE_SET

#
=====
# api.proxy
# Use the following properties for communication
# with the Contrast UI over a proxy.
#
=====
# proxy:

# Set value to `true` for the agent to communicate with
# the Contrast web interface over a proxy. Set value to
# `false` if you don't want to use the proxy. If no value is
# indicated, the presence of a valid **contrast.proxy.host**
# and **contrast.proxy.port** will enable the proxy.
# enable: NEEDS_TO_BE_SET

# Set the proxy host. It must be set with port and scheme.
# host: localhost

# Set the proxy port. It must be set with host and scheme.
# port: 1234

# Set the proxy scheme (e.g., `http` or
# `https`). It must be set with host and port.
# scheme: http

# Set this property as an alternate for `scheme://host:port`. It takes
# precedence over the other settings, if specified; however, an error
```

```
# will be thrown if both the URL and individual properties are set.
# url: NEEDS_TO_BE_SET

# Set the proxy user.
# user: NEEDS_TO_BE_SET

# Set the proxy password.
# pass: NEEDS_TO_BE_SET

# Set the proxy authentication type. Value
# options are `NTLM`, `Digest`, and `Basic`.
# auth_type: NEEDS_TO_BE_SET

#
=====
==
# agent
# Use the properties in this section to control the way and frequency
# with which the agent communicates to logs and the Contrast UI.
#
=====
==
# agent:

#
=====
# agent.diagnostics
# Use the properties in this section to specify the information the agent
# should collect and report in order to diagnose problems in the agent.
#
#
=====
# diagnostics:

# Creates config and system info files
# at startup if true. True by default.
#
# The same thing can be achieved by setting the
# CONTRAST__AGENT__DIAGNOSTICS__ENABLE=[true/false] env variable.
#
# enable: true

#
=====
# agent.route_coverage
# Use the following properties for the route-based coverage feature.
#
=====
# route_coverage: {}

#
=====
# agent.reporting
# Use the following settings to configure reporting to the Contrast UI.
#
```

```
=====
# reporting:

# Set the grace period (in milliseconds) after
# agent shutdown to allow draining pending reports.
# shutdown_grace_period_ms: 120000

#
=====
# agent.effective_config
# None
#
=====
# effective_config:

#
=====
# agent.effective_config.reporting
# None
#
=====
# reporting:

# Defaults to `true`. Controls whether configuration
# setting reports are sent to the Contrast web interface.
# enable: true

#
=====
# agent.logger
# Define the following properties to set logging values.
# If the following properties are not defined, the
# agent uses the logging values from the Contrast UI.
#
=====
# logger:

# Enable diagnostic logging by setting a path to a log file.
# While diagnostic logging hurts performance, it generates
# useful information for debugging Contrast. The value set here
# is the location to which the agent saves log output. If no
# log file exists at this location, the agent creates a file.
#
# Example - `/opt/Contrast/contrast.log` creates a log in the
# `/opt/Contrast` directory, and rotates it automatically as needed.
#
# path: ./contrast_agent.log

# Set the the log output level. Valid options are
# `ERROR`, `WARN`, `INFO`, `DEBUG`, and `TRACE`.
# level: INFO

# Set to `true` to redirect all logs to
# `stdout` instead of the file system.
# stdout: false
```

```
# Set to `true` to redirect all logs to `stderr` instead of
# the file system. May be combined with the corresponding
# `stdout` configuration to write to both streams.
# stderr: false

# Change the Contrast logger from a file-sized based rolling scheme
# to a date-based rolling scheme. At midnight server time, the
# previous day log is renamed to *file_name.yyyy-MM-dd*. Note -
# this scheme does not have a size limit; manual log pruning is
# required. You must set this flag to use the backups and size flags.
# roll_daily: false

# Set the roll size for log files in megabytes. The agent will
# attempt to prevent the log file from being larger than this size.
# roll_size: 100

# Set the number of backup files to keep. Set to `0` to disable.
# backups: 10

#
=====
# agent.security_logger
# Define the following properties to set security logging
# values associated with Protect. If not defined, the agent
# uses the security logging (CEF) values from the Contrast UI.
#
=====
# security_logger:

# Set the file to which the agent logs security events.
# path: ./contrast/security.log

# Set the log level for security logging. Valid options
# are `ERROR`, `WARN`, `INFO`, `DEBUG`, and `TRACE`.
# level: ERROR

# Change the Contrast security logger from a file-sized based rolling
# scheme to a date-based rolling scheme. At midnight server time,
# the log from the previous day is renamed to *file_name.yyyy-MM-dd*.
# Note - this scheme does not have a size limit; manual log
# pruning will be required. This flag must be set to use the
# backups and size flags. Value options are `true` or `false`.
# roll_daily: NEEDS_TO_BE_SET

# Specify the file size cap (in MB) of each log file.
# roll_size: NEEDS_TO_BE_SET

# Specify the number of backup logs that the agent will create before
# Contrast cleans up the oldest file. A value of `0` means that no
backups
# are created, and the log is truncated when it reaches its size cap.
#
# Note - this property must be used with
# `agent.security_logger.roll_daily=false`; otherwise,
```

```
# Contrast continues to log daily and disregard this limit.
#
# backups: NEEDS_TO_BE_SET

#
=====
# agent.security_logger.syslog
# Define the following properties to set Syslog values. If the
properties
# are not defined, the agent uses the Syslog values from the Contrast
UI.
#
=====
# syslog:

# Set to `true` to enable Syslog logging.
# enable: NEEDS_TO_BE_SET

# Set the IP address of the Syslog server
# to which the agent should send messages.
# ip: NEEDS_TO_BE_SET

# Set the port of the Syslog server to
# which the agent should send messages.
# port: NEEDS_TO_BE_SET

# Set the facility code of the messages the agent sends to Syslog.
# facility: 19

# Set the log level of Exploited attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_exploited: ALERT

# Set the log level of Blocked attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_blocked: NOTICE

# Set the log level of Blocked At Perimeter
# attacks. Value options are `ALERT`, `CRITICAL`,
# `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_blocked_perimeter: NOTICE

# Set the log level of Probed attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_probed: WARNING

# Set the log level of Suspicious attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_suspicious: WARNING

#
=====
# agent.security_logger.syslog.heartbeat
# Define the following properties to
# set the Syslog heartbeat properties.
```

```
#
=====
# heartbeat:

# Set to `true` to enable the Syslog heartbeat.
# The heartbeat will issue a Syslog message at
# the INFO level after every interval passes.
# enable: false

# Set the interval for sending heartbeat messages
# to the Syslog server (in milliseconds).
# interval_ms: 60000

#
=====
# agent.java
# The following properties apply to any Java agent-wide configurations.
#
=====
# java:

# Configure the Java agent to skip its application discovery
# algorithm, and instead associate all libraries, vulnerabilities,
# and web traffic to a single application with the name specified
# by this property. This configuration is preferred when deploying
# Java SE applications with embedded web servers (e.g., applications
# built with Spring Boot, Dropwizard, and embedded Jetty). When used
# with an application server, this configuration associates all
# web traffic with the single, standalone application, including
# web traffic handled by application server-hosted endpoints that
# would not be associated with a discovered application otherwise.
#
# Note - This settings takes preferences
# over the `application.name` setting.
#
# standalone_app_name: NEEDS_TO_BE_SET

# By default, the Java agent visits all classes at startup to look
# for vulnerabilities, which the agent may detect by scanning a
# class (e.g., hardcoded passwords). Set this property to `false`
# to disable the default behavior. If disabled, the agent will
# only visit classes which are likely to require sensors; this
# can improve application startup time, but may produce fewer
# findings (most likely findings that require static analysis).
#
# scan_all_classes: true

# By default, the Java agent deeply inspects all JAR and WAR files
loaded
# by the JVM to build a comprehensive understanding of the type
hierarchy.
# This understanding allows Contrast to instrument sensors into types
# that it might have overlooked. In most cases, this produces a slight
# increase in accuracy at the cost of increased application startup
# time. Set this property to `false` to disable this level of
```

```
inspection.
#
# scan_all_code_sources: true

#
=====
==
# inventory
# Use the properties in this section to override the inventory features.
#
=====
==
# inventory:

# Set to `false` to disable inventory features in the agent.
# enable: true

# Define a list of directories where libraries are stored.
# Directories must be formatted as a semicolon-delimited
# list for Windows or a colon-delimited list for Linux.
#
# Define a list of directories where libraries are stored.
# Directories must be formatted as a semicolon-delimited list.
# Example - `path1;path2;path3`(Windows) or `path1:path2:path3`(Linux)
# Example - `path1;path2;path3`
#
# library_dirs: NEEDS_TO_BE_SET

# Set the maximum archive unpacking depth when analyzing libraries.
# library_depth: 10

# Set the boolean to more aggressively limit the
# manifest information reported for libraries. If true,
# the limit is 1,000 characters, otherwise it's 3,000.
# prune_package_details: true

# Apply a list of labels to libraries. Labels
# must be formatted as a comma-delimited list.
# Example - `label1, label2, label3`
#
# tags: NEEDS_TO_BE_SET

#
=====
==
# assess
# Use the properties in this section to control Assess.
#
=====
==
# assess:

# Include this property to determine if the Assess
# feature should be enabled. If this property is not
# present, the decision is delegated to the Contrast UI.
```



```
# enable: false

# Control the values captured by Assess vulnerability events. `Full`
# captures most values by calling ToString on objects, which can
# provide more info but causes increased memory usage. `Minimal`
# has better performance as it only captures String type objects
# as strings and uses type name for other object type values.
# event_detail: minimal

# Apply a list of labels to vulnerabilities and preflight
# messages. Labels must be formatted as a comma-delimited list.
# Example - `label1, label2, label3`
#
# tags: NEEDS_TO_BE_SET

#
=====
# assess.sampling
# Use the following properties to control sampling in the agent.
#
=====
# sampling:

# Set to `true` to enable sampling.
# enable: false

# This property indicates the number of requests
# to analyze in each window before sampling begins.
# baseline: 5

# This property indicates that every *nth*
# request after the baseline is analyzed.
# request_frequency: 10

# This property indicates the duration for which a sample set is valid.
# window_ms: 180_000

#
=====
# assess.rules
# Use the following properties to control simple rule configurations.
#
=====
# rules:

# Define a list of Assess rules to disable in the agent. To view a
# list of rule names, in Contrast go to user menu > Policy Management >
# Assess rules. The rules must be formatted as a comma-delimited list.
#
# Example - Set `reflected-xss,sql-injection` to disable
# the reflected-xss rule and the sql-injection rule.
#
# disabled_rules: NEEDS_TO_BE_SET

#
```

```
=====
==
# profile
# Set configuration values under a profile name to enable
# multi-tenant application configuration on web servers. See
# https://support.contrastsecurity.com/hc/en-us/articles/360052187171-Multi-Application-configuration-with-Contrast-Profiles
# for more details.
#
=====
==
# profile: {}

#
=====
==
# protect
# Use the properties in this section to override Protect features.
#
=====
==
# protect:

# Include this property to determine if the Protect
# feature should be enabled. If this property is not
# present, the decision is delegated to the Contrast UI.
# enable: false

#
=====
# protect.rules
# Use the following properties to set simple rule configurations.
#
=====
# rules:

# Define a list of Protect rules to disable in the agent. To view a
# list of rule names, in Contrast go to user menu > Policy Management >
# Protect rules. The rules must be formatted as a comma-delimited list.
# disabled_rules: NEEDS_TO_BE_SET

#
=====
# protect.rules.bot-blocker
# Use the following selection to configure if the
# agent blocks bots. Set to `true` to enable blocking.
#
=====
# bot-blocker:

# Set to `true` for the agent to block known bots.
# enable: false

#
=====
```

```
# protect.rules.sql-injection
# Use the following settings to configure the sql-injection rule.
#
=====
# sql-injection:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or off.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

# Tell the agent to detect when semantic analysis of the query
# reveals tautologies used in exfiltration attacks (e.g., "or
# 1=1" or "or 2<>3"). The agent blocks if blocking is enabled.
# detect_tautologies: false

# Tell the agent to detect when semantic analysis of the query
# reveals the invocation of dangerous functions typically used in
# weaponized exploits. The agent blocks if blocking is enabled.
# detect_dangerous_functions: false

# Tell the agent to detect when semantic analysis of the query
# reveals chained queries, which is uncommon in normal usage but
# common in exploit. The agent blocks if blocking is enabled.
# detect_chained_queries: false

# Tell the agent to detect when semantic analysis of the query
# reveals database queries are being made for system tables and
# sensitive information. The agent blocks if blocking is enabled.
# detect_suspicious_unions: false

# Tell the agent to be more aggressive in detecting user
# inputs as SQL comments. This enables the agent to better
# detect SQL Injection input vectors that use comments to
# terminate queries. The agent blocks if blocking is enabled.
# aggressive_comment: false

#
=====
# protect.rules.cmd-injection
# Use the following properties to configure
# how the command injection rule works.
#
=====
# cmd-injection:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
```

```
# mode: off

# Detect when the agent sees user parameters being executed as
# system commands. The agent blocks if blocking is enabled.
# detect_parameter_command_backdoors: true

# Detect when a system command is issued which contains
# chained commands. The agent blocks if blocking is enabled.
# detect_chained_commands: true

# Detect when a system command is issued with an argument matching a
# known dangerous file path. The agent blocks if blocking is enabled.
# detect_dangerous_path_args: true

# Tell the agent to detect when commands come directly
# from input. The agent blocks if blocking is enabled.
# detect_phased_commands: true

#
=====
# protect.rules.cmd-injection-process-hardening
# Use the following settings to configure whether
# the agent blocks all attempts to start an external
# process. To enable blocking, set to 'true'.
#
=====
# cmd-injection-process-hardening:

# Set to `true` to enable the agent to block
# all attempts to start external processes.
# enable: false

#
=====
# protect.rules.path-traversal
# Use the following properties to configure
# how the path traversal rule works.
#
=====
# path-traversal:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

# Detect when custom code attempts to access sensitive
# system files. The agent blocks if blocking is enabled.
# detect_custom_code_accessing_system_files: true

# Detect when users attempt to bypass filters by
# using ":::$DATA" channels or null bytes in file
```

```
# names. The agent blocks if blocking is enabled.
# detect_common_file_exploits: true

#
=====
# protect.rules.method-tampering
# Use the following properties to configure
# how the method tampering rule works.
#
=====
# method-tampering:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.reflected-xss
# Use the following properties to configure how
# the reflected cross-site scripting rule works.
#
=====
# reflected-xss:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.xxe
# Use the following properties to configure
# how the XML external entity works.
#
=====
# xxe:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
```

```
=====
# protect.rules.padding-oracle
# Use the following properties to configure
# how the padding-oracle rule works.
#
=====
# padding-oracle: {}

#
=====
==
# application
# Use the properties in this section for
# the application(s) hosting this agent.
#
=====
==
# application:

# Override the reported application name.
#
# Note - On systems where multiple, distinct applications may be served
# by a single process, this configuration causes the agent to report
# all discovered applications as one application with the given name.
#
# Note - On Java systems where multiple, distinct applications may be
# served by a single process, this configuration causes the agent to
report
# all discovered applications as one application with the given name.
#
# name: NEEDS_TO_BE_SET

# Override the reported application path.
# path: NEEDS_TO_BE_SET

# Add the name of the application group with which this
# application should be associated in the Contrast UI.
# group: NEEDS_TO_BE_SET

# Add the application code this application should use in the Contrast UI.
# code: NEEDS_TO_BE_SET

# Override the reported application version.
# version: NEEDS_TO_BE_SET

# Apply labels to an application. Labels must
# be formatted as a comma-delimited list.
# Example - `label1,label2,label3`
#
# tags: NEEDS_TO_BE_SET

# Define a set of `key=value` pairs (which conforms to RFC 2253) for
# specifying user-defined metadata associated with the application. The
# set must be formatted as a comma-delimited list of `key=value` pairs.
# Example - `business-unit=accounting, office=Baltimore`
```

```
#
# metadata: NEEDS_TO_BE_SET

# Provide the ID of a session which already exists in the Contrast
# UI. Vulnerabilities discovered by the agent are associated with
# this session. If an invalid ID is supplied, the agent will be
# disabled. This option and `application.session_metadata` are
# mutually exclusive; if both are set, the agent will be disabled.
# session_id: NEEDS_TO_BE_SET

# Provide metadata which is used to create a new session ID in the
# Contrast UI. Vulnerabilities discovered by the agent are associated with
# this new session. This value should be formatted as `key=value` pairs
# (conforming to RFC 2253). Available key names for this configuration
# are branchName, buildNumber, commitHash, committer, gitTag, repository,
# testRun, and version. This option and `application.session_id` are
# mutually exclusive; if both are set the agent will be disabled.
# session_metadata: NEEDS_TO_BE_SET

#
=====
==
# server
# Use the settings in this section to set metadata for the server
# hosting this agent. Contrast recognizes common, supported server
# names, paths, types and environments. Doing this may require a new
# server or license, and it may affect functionality of some features.
#
=====
==
# server:

# Override the reported server name.
# name: localhost

# Override the reported server path.
# path: NEEDS_TO_BE_SET

# Override the reported server type.
# type: NEEDS_TO_BE_SET

# Set the environment directly to override the default set
# by the Contrast UI. This allows the user to configure the
# environment dynamically at startup rather than manually
# updating the Server in the Contrast UI themselves afterwards.
#
# Valid values include `QA`, `PRODUCTION` and `DEVELOPMENT`.
# For example, `PRODUCTION` registers this Server as
# running in a `PRODUCTION` environment, regardless of the
# organization's default environment in the Contrast UI.
#
# environment: NEEDS_TO_BE_SET

# Apply a list of labels to the server. Labels
# must be formatted as a comma-delimited list.
```

```
# Example - `label1,label2,label3`
#
# tags: NEEDS_TO_BE_SET

# Set to `false` to disable detection of cloud
# provider metadata such as resource identifiers.
# discover_cloud_resource: true
```

Java system properties

Substitute <YourContrastJarPath> with the path to your [Contrast JAR \(page 120\)](#), and use these commands to learn more about system properties:

- To generate a list of general properties using the Contrast agent JAR, use:

```
java -jar <YourContrastJarPath> properties
```

- Use command line with tools to search for commands. For example, these commands display a list of proxy-related properties:

Using the built-in filter:

```
java -jar <YourContrastJarPath> properties --filter=proxy
```

Java YAML configuration template

Use this template to configure the Java agent using a YAML configuration file. (Learn more about [YAML configuration \(page 107\)](#).)

Place your YAML file in the default location:

- Unix:** /etc/contrast/java/contrast_security.yaml
- Windows:** C:/ProgramData/contrast/java/contrast_security.yaml

```
#
=====
==
# Use the properties in this YAML file to configure a Contrast agent.
# Go to https://docs.contrastsecurity.com/en/order-of-precedence.html
# to determine the order of precedence for configuration values.
#
=====
==

# Use this setting if you want to temporarily disable a Contrast agent.
# Set to `true` to enable the agent; set to `false` to disable the agent.
# enable: true

#
=====
==
# api
# Use the properties in this section to connect the agent to the Contrast
UI.
#
=====
==
api:
```



```
# ***** REQUIRED *****
# Set the URL for the Contrast UI.
url: https://app.contrastsecurity.com/Contrast

# ***** REQUIRED *****
# Set the API key needed to communicate with the Contrast UI.
api_key: NEEDS_TO_BE_SET

# ***** REQUIRED *****
# Set the service key needed to communicate with the Contrast
# UI. It is used to calculate the Authorization header.
service_key: NEEDS_TO_BE_SET

# ***** REQUIRED *****
# Set the user name used to communicate with the Contrast
# UI. It is used to calculate the Authorization header.
user_name: NEEDS_TO_BE_SET

# base64 encoded JSON object containing the `url`,
# `api_key`, `service_key`, and `user_name` config options,
# allowing them all to be set in a single variable.
# token: NEEDS_TO_BE_SET

# Set the timeout for communicating with TeamServer. This property will be
# respected over the deprecated legacy configuration *contrast.timeout*.
# timeout_ms: NEEDS_TO_BE_SET

#
=====
# api.proxy
# Use the following properties for communication
# with the Contrast UI over a proxy.
#
=====
# proxy:

# Set value to `true` for the agent to communicate with
# the Contrast web interface over a proxy. Set value to
# `false` if you don't want to use the proxy. If no value is
# indicated, the presence of a valid **contrast.proxy.host**
# and **contrast.proxy.port** will enable the proxy.
# enable: NEEDS_TO_BE_SET

# Set the proxy host. It must be set with port and scheme.
# host: localhost

# Set the proxy port. It must be set with host and scheme.
# port: 1234

# Set the proxy scheme (e.g., `http` or
# `https`). It must be set with host and port.
# scheme: http

# Set this property as an alternate for `scheme://host:port`. It takes
```

```
# precedence over the other settings, if specified; however, an error
# will be thrown if both the URL and individual properties are set.
# url: NEEDS_TO_BE_SET

# Set the proxy user.
# user: NEEDS_TO_BE_SET

# Set the proxy password.
# pass: NEEDS_TO_BE_SET

# Set the proxy authentication type. Value
# options are `NTLM`, `Digest`, and `Basic`.
# auth_type: NEEDS_TO_BE_SET

#
=====
==
# agent
# Use the properties in this section to control the way and frequency
# with which the agent communicates to logs and the Contrast UI.
#
=====
==
# agent:

#
=====
# agent.diagnostics
# Use the properties in this section to specify the information the agent
# should collect and report in order to diagnose problems in the agent.
#
#
=====
# diagnostics:

# Creates config and system info files
# at startup if true. True by default.
#
# The same thing can be achieved by setting the
# CONTRAST__AGENT__DIAGNOSTICS__ENABLE=[true/false] env variable.
#
# enable: true

#
=====
# agent.route_coverage
# Use the following properties for the route-based coverage feature.
#
=====
# route_coverage: {}

#
=====
# agent.reporting
# Use the following settings to configure reporting to the Contrast UI.
```

```
#
=====
# reporting:

# Set the grace period (in milliseconds) after
# agent shutdown to allow draining pending reports.
# shutdown_grace_period_ms: 120000

#
=====
# agent.effective_config
# None
#
=====
# effective_config:

#
=====
# agent.effective_config.reporting
# None
#
=====
# reporting:

# Defaults to `true`. Controls whether configuration
# setting reports are sent to the Contrast web interface.
# enable: true

#
=====
# agent.logger
# Define the following properties to set logging values.
# If the following properties are not defined, the
# agent uses the logging values from the Contrast UI.
#
=====
# logger:

# Enable diagnostic logging by setting a path to a log file.
# While diagnostic logging hurts performance, it generates
# useful information for debugging Contrast. The value set here
# is the location to which the agent saves log output. If no
# log file exists at this location, the agent creates a file.
#
# Example - `/opt/Contrast/contrast.log` creates a log in the
# `/opt/Contrast` directory, and rotates it automatically as needed.
#
# path: ./contrast_agent.log

# Set the the log output level. Valid options are
# `ERROR`, `WARN`, `INFO`, `DEBUG`, and `TRACE`.
# level: INFO

# Set to `true` to redirect all logs to
# `stdout` instead of the file system.
```

```
# stdout: false

# Set to `true` to redirect all logs to `stderr` instead of
# the file system. May be combined with the corresponding
# `stdout` configuration to write to both streams.
# stderr: false

# Change the Contrast logger from a file-sized based rolling scheme
# to a date-based rolling scheme. At midnight server time, the
# previous day log is renamed to *file_name.yyyy-MM-dd*. Note -
# this scheme does not have a size limit; manual log pruning is
# required. You must set this flag to use the backups and size flags.
# roll_daily: false

# Set the roll size for log files in megabytes. The agent will
# attempt to prevent the log file from being larger than this size.
# roll_size: 100

# Set the number of backup files to keep. Set to `0` to disable.
# backups: 10

#
=====
# agent.security_logger
# Define the following properties to set security logging
# values associated with Protect. If not defined, the agent
# uses the security logging (CEF) values from the Contrast UI.
#
=====
# security_logger:

# Set the file to which the agent logs security events.
# path: ./contrast/security.log

# Set the log level for security logging. Valid options
# are `ERROR`, `WARN`, `INFO`, `DEBUG`, and `TRACE`.
# level: ERROR

# Change the Contrast security logger from a file-sized based rolling
# scheme to a date-based rolling scheme. At midnight server time,
# the log from the previous day is renamed to *file_name.yyyy-MM-dd*.
# Note - this scheme does not have a size limit; manual log
# pruning will be required. This flag must be set to use the
# backups and size flags. Value options are `true` or `false`.
# roll_daily: NEEDS_TO_BE_SET

# Specify the file size cap (in MB) of each log file.
# roll_size: NEEDS_TO_BE_SET

# Specify the number of backup logs that the agent will create before
# Contrast cleans up the oldest file. A value of `0` means that no
backups
# are created, and the log is truncated when it reaches its size cap.
#
# Note - this property must be used with
```

```
# `agent.security_logger.roll_daily=false`; otherwise,
# Contrast continues to log daily and disregard this limit.
#
# backups: NEEDS_TO_BE_SET

#
=====
# agent.security_logger.syslog
# Define the following properties to set Syslog values. If the
properties
# are not defined, the agent uses the Syslog values from the Contrast
UI.
#
=====
# syslog:

# Set to `true` to enable Syslog logging.
# enable: NEEDS_TO_BE_SET

# Set the IP address of the Syslog server
# to which the agent should send messages.
# ip: NEEDS_TO_BE_SET

# Set the port of the Syslog server to
# which the agent should send messages.
# port: NEEDS_TO_BE_SET

# Set the facility code of the messages the agent sends to Syslog.
# facility: 19

# Set the log level of Exploited attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_exploited: ALERT

# Set the log level of Blocked attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_blocked: NOTICE

# Set the log level of Blocked At Perimeter
# attacks. Value options are `ALERT`, `CRITICAL`,
# `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_blocked_perimeter: NOTICE

# Set the log level of Probed attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_probed: WARNING

# Set the log level of Suspicious attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_suspicious: WARNING

#
=====
# agent.security_logger.syslog.heartbeat
# Define the following properties to
```

```
# set the Syslog heartbeat properties.
#
=====
# heartbeat:

# Set to `true` to enable the Syslog heartbeat.
# The heartbeat will issue a Syslog message at
# the INFO level after every interval passes.
# enable: false

# Set the interval for sending heartbeat messages
# to the Syslog server (in milliseconds).
# interval_ms: 60000

#
=====
# agent.java
# The following properties apply to any Java agent-wide configurations.
#
=====
# java:

# Configure the Java agent to skip its application discovery
# algorithm, and instead associate all libraries, vulnerabilities,
# and web traffic to a single application with the name specified
# by this property. This configuration is preferred when deploying
# Java SE applications with embedded web servers (e.g., applications
# built with Spring Boot, Dropwizard, and embedded Jetty). When used
# with an application server, this configuration associates all
# web traffic with the single, standalone application, including
# web traffic handled by application server-hosted endpoints that
# would not be associated with a discovered application otherwise.
#
# Note - This settings takes preferences
# over the `application.name` setting.
#
# standalone_app_name: NEEDS_TO_BE_SET

# By default, the Java agent visits all classes at startup to look
# for vulnerabilities, which the agent may detect by scanning a
# class (e.g., hardcoded passwords). Set this property to `false`
# to disable the default behavior. If disabled, the agent will
# only visit classes which are likely to require sensors; this
# can improve application startup time, but may produce fewer
# findings (most likely findings that require static analysis).
#
# scan_all_classes: true

# By default, the Java agent deeply inspects all JAR and WAR files
loaded
# by the JVM to build a comprehensive understanding of the type
hierarchy.
# This understanding allows Contrast to instrument sensors into types
# that it might have overlooked. In most cases, this produces a slight
# increase in accuracy at the cost of increased application startup
```

```
# time. Set this property to `false` to disable this level of
inspection.
#
# scan_all_code_sources: true

#
=====
==
# inventory
# Use the properties in this section to override the inventory features.
#
=====
==
# inventory:

# Set to `false` to disable inventory features in the agent.
# enable: true

# Define a list of directories where libraries are stored.
# Directories must be formatted as a semicolon-delimited
# list for Windows or a colon-delimited list for Linux.
#
# Define a list of directories where libraries are stored.
# Directories must be formatted as a semicolon-delimited list.
# Example - `path1;path2;path3` (Windows) or `path1:path2:path3` (Linux)
# Example - `path1;path2;path3`
#
# library_dirs: NEEDS_TO_BE_SET

# Set the maximum archive unpacking depth when analyzing libraries.
# library_depth: 10

# Set the boolean to more aggressively limit the
# manifest information reported for libraries. If true,
# the limit is 1,000 characters, otherwise it's 3,000.
# prune_package_details: true

# Apply a list of labels to libraries. Labels
# must be formatted as a comma-delimited list.
# Example - `label1, label2, label3`
#
# tags: NEEDS_TO_BE_SET

#
=====
==
# assess
# Use the properties in this section to control Assess.
#
=====
==
# assess:

# Include this property to determine if the Assess
# feature should be enabled. If this property is not
```

```
# present, the decision is delegated to the Contrast UI.
# enable: false

# Control the values captured by Assess vulnerability events. `Full`
# captures most values by calling ToString on objects, which can
# provide more info but causes increased memory usage. `Minimal`
# has better performance as it only captures String type objects
# as strings and uses type name for other object type values.
# event_detail: minimal

# Apply a list of labels to vulnerabilities and preflight
# messages. Labels must be formatted as a comma-delimited list.
# Example - `label1, label2, label3`
#
# tags: NEEDS_TO_BE_SET

#
=====
# assess.sampling
# Use the following properties to control sampling in the agent.
#
=====
# sampling:

# Set to `true` to enable sampling.
# enable: false

# This property indicates the number of requests
# to analyze in each window before sampling begins.
# baseline: 5

# This property indicates that every *nth*
# request after the baseline is analyzed.
# request_frequency: 10

# This property indicates the duration for which a sample set is valid.
# window_ms: 180_000

#
=====
# assess.rules
# Use the following properties to control simple rule configurations.
#
=====
# rules:

# Define a list of Assess rules to disable in the agent. To view a
# list of rule names, in Contrast go to user menu > Policy Management >
# Assess rules. The rules must be formatted as a comma-delimited list.
#
# Example - Set `reflected-xss,sql-injection` to disable
# the reflected-xss rule and the sql-injection rule.
#
# disabled_rules: NEEDS_TO_BE_SET
```



```
#
=====
==
# profile
# Set configuration values under a profile name to enable
# multi-tenant application configuration on web servers. See
# https://support.contrastsecurity.com/hc/en-us/articles/360052187171-Multi-Application-configuration-with-Contrast-Profiles
# for more details.
#
=====
==
# profile: {}

#
=====
==
# protect
# Use the properties in this section to override Protect features.
#
=====
==
# protect:

# Include this property to determine if the Protect
# feature should be enabled. If this property is not
# present, the decision is delegated to the Contrast UI.
# enable: false

#
=====
# protect.rules
# Use the following properties to set simple rule configurations.
#
=====
# rules:

# Define a list of Protect rules to disable in the agent. To view a
# list of rule names, in Contrast go to user menu > Policy Management >
# Protect rules. The rules must be formatted as a comma-delimited list.
# disabled_rules: NEEDS_TO_BE_SET

#
=====
# protect.rules.bot-blocker
# Use the following selection to configure if the
# agent blocks bots. Set to `true` to enable blocking.
#
=====
# bot-blocker:

# Set to `true` for the agent to block known bots.
# enable: false

#
```

```
=====
# protect.rules.sql-injection
# Use the following settings to configure the sql-injection rule.
#
=====
# sql-injection:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or off.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

# Tell the agent to detect when semantic analysis of the query
# reveals tautologies used in exfiltration attacks (e.g., "or
# 1=1" or "or 2<>3"). The agent blocks if blocking is enabled.
# detect_tautologies: false

# Tell the agent to detect when semantic analysis of the query
# reveals the invocation of dangerous functions typically used in
# weaponized exploits. The agent blocks if blocking is enabled.
# detect_dangerous_functions: false

# Tell the agent to detect when semantic analysis of the query
# reveals chained queries, which is uncommon in normal usage but
# common in exploit. The agent blocks if blocking is enabled.
# detect_chained_queries: false

# Tell the agent to detect when semantic analysis of the query
# reveals database queries are being made for system tables and
# sensitive information. The agent blocks if blocking is enabled.
# detect_suspicious_unions: false

# Tell the agent to be more aggressive in detecting user
# inputs as SQL comments. This enables the agent to better
# detect SQL Injection input vectors that use comments to
# terminate queries. The agent blocks if blocking is enabled.
# aggressive_comment: false

#
=====
# protect.rules.cmd-injection
# Use the following properties to configure
# how the command injection rule works.
#
=====
# cmd-injection:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
```

```
#
# mode: off

# Detect when the agent sees user parameters being executed as
# system commands. The agent blocks if blocking is enabled.
# detect_parameter_command_backdoors: true

# Detect when a system command is issued which contains
# chained commands. The agent blocks if blocking is enabled.
# detect_chained_commands: true

# Detect when a system command is issued with an argument matching a
# known dangerous file path. The agent blocks if blocking is enabled.
# detect_dangerous_path_args: true

# Tell the agent to detect when commands come directly
# from input. The agent blocks if blocking is enabled.
# detect_phased_commands: true

#
=====
# protect.rules.cmd-injection-process-hardening
# Use the following settings to configure whether
# the agent blocks all attempts to start an external
# process. To enable blocking, set to 'true'.
#
=====
# cmd-injection-process-hardening:

# Set to `true` to enable the agent to block
# all attempts to start external processes.
# enable: false

#
=====
# protect.rules.path-traversal
# Use the following properties to configure
# how the path traversal rule works.
#
=====
# path-traversal:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

# Detect when custom code attempts to access sensitive
# system files. The agent blocks if blocking is enabled.
# detect_custom_code_accessing_system_files: true

# Detect when users attempt to bypass filters by
```

```
# using "::$DATA" channels or null bytes in file
# names. The agent blocks if blocking is enabled.
# detect_common_file_exploits: true

#
=====
# protect.rules.method-tampering
# Use the following properties to configure
# how the method tampering rule works.
#
=====
# method-tampering:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.reflected-xss
# Use the following properties to configure how
# the reflected cross-site scripting rule works.
#
=====
# reflected-xss:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.xxe
# Use the following properties to configure
# how the XML external entity works.
#
=====
# xxe:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off
```

```
#
=====
# protect.rules.padding-oracle
# Use the following properties to configure
# how the padding-oracle rule works.
#
=====
# padding-oracle: {}

#
=====
==
# application
# Use the properties in this section for
# the application(s) hosting this agent.
#
=====
==
# application:

# Override the reported application name.
#
# Note - On systems where multiple, distinct applications may be served
# by a single process, this configuration causes the agent to report
# all discovered applications as one application with the given name.
#
# Note - On Java systems where multiple, distinct applications may be
# served by a single process, this configuration causes the agent to
report
# all discovered applications as one application with the given name.
#
# name: NEEDS_TO_BE_SET

# Override the reported application path.
# path: NEEDS_TO_BE_SET

# Add the name of the application group with which this
# application should be associated in the Contrast UI.
# group: NEEDS_TO_BE_SET

# Add the application code this application should use in the Contrast UI.
# code: NEEDS_TO_BE_SET

# Override the reported application version.
# version: NEEDS_TO_BE_SET

# Apply labels to an application. Labels must
# be formatted as a comma-delimited list.
# Example - `label1,label2,label3`
#
# tags: NEEDS_TO_BE_SET

# Define a set of `key=value` pairs (which conforms to RFC 2253) for
# specifying user-defined metadata associated with the application. The
# set must be formatted as a comma-delimited list of `key=value` pairs.
```

```
# Example - `business-unit=accounting, office=Baltimore`
#
# metadata: NEEDS_TO_BE_SET

# Provide the ID of a session which already exists in the Contrast
# UI. Vulnerabilities discovered by the agent are associated with
# this session. If an invalid ID is supplied, the agent will be
# disabled. This option and `application.session_metadata` are
# mutually exclusive; if both are set, the agent will be disabled.
# session_id: NEEDS_TO_BE_SET

# Provide metadata which is used to create a new session ID in the
# Contrast UI. Vulnerabilities discovered by the agent are associated with
# this new session. This value should be formatted as `key=value` pairs
# (conforming to RFC 2253). Available key names for this configuration
# are branchName, buildNumber, commitHash, committer, gitTag, repository,
# testRun, and version. This option and `application.session_id` are
# mutually exclusive; if both are set the agent will be disabled.
# session_metadata: NEEDS_TO_BE_SET

#
=====
==
# server
# Use the settings in this section to set metadata for the server
# hosting this agent. Contrast recognizes common, supported server
# names, paths, types and environments. Doing this may require a new
# server or license, and it may affect functionality of some features.
#
=====
==
# server:

# Override the reported server name.
# name: localhost

# Override the reported server path.
# path: NEEDS_TO_BE_SET

# Override the reported server type.
# type: NEEDS_TO_BE_SET

# Set the environment directly to override the default set
# by the Contrast UI. This allows the user to configure the
# environment dynamically at startup rather than manually
# updating the Server in the Contrast UI themselves afterwards.
#
# Valid values include `QA`, `PRODUCTION` and `DEVELOPMENT`.
# For example, `PRODUCTION` registers this Server as
# running in a `PRODUCTION` environment, regardless of the
# organization's default environment in the Contrast UI.
#
# environment: NEEDS_TO_BE_SET

# Apply a list of labels to the server. Labels
```

```
# must be formatted as a comma-delimited list.
# Example - `label1,label2,label3`
#
# tags: NEEDS_TO_BE_SET

# Set to `false` to disable detection of cloud
# provider metadata such as resource identifiers.
# discover_cloud_resource: true
```

```
#
=====
==
# Use the properties in this YAML file to configure a Contrast agent.
# Go to https://docs.contrastsecurity.com/en/order-of-precedence.html
# to determine the order of precedence for configuration values.
#
=====
==

# Use this setting if you want to temporarily disable a Contrast agent.
# Set to `true` to enable the agent; set to `false` to disable the agent.
# enable: true

#
=====
==
# api
# Use the properties in this section to connect the agent to the Contrast
# UI.
#
=====
==
api:

# ***** REQUIRED *****
# Set the URL for the Contrast UI.
url: https://app.contrastsecurity.com/Contrast

# ***** REQUIRED *****
# Set the API key needed to communicate with the Contrast UI.
api_key: NEEDS_TO_BE_SET

# ***** REQUIRED *****
# Set the service key needed to communicate with the Contrast
# UI. It is used to calculate the Authorization header.
service_key: NEEDS_TO_BE_SET

# ***** REQUIRED *****
# Set the user name used to communicate with the Contrast
# UI. It is used to calculate the Authorization header.
user_name: NEEDS_TO_BE_SET

# base64 encoded JSON object containing the `url`,
# `api_key`, `service_key`, and `user_name` config options,
```

```
# allowing them all to be set in a single variable.
# token: NEEDS_TO_BE_SET

# Set the timeout for communicating with TeamServer. This property will be
# respected over the deprecated legacy configuration *contrast.timeout*.
# timeout_ms: NEEDS_TO_BE_SET

#
=====
# api.proxy
# Use the following properties for communication
# with the Contrast UI over a proxy.
#
=====
# proxy:

# Set value to `true` for the agent to communicate with
# the Contrast web interface over a proxy. Set value to
# `false` if you don't want to use the proxy. If no value is
# indicated, the presence of a valid **contrast.proxy.host**
# and **contrast.proxy.port** will enable the proxy.
# enable: NEEDS_TO_BE_SET

# Set the proxy host. It must be set with port and scheme.
# host: localhost

# Set the proxy port. It must be set with host and scheme.
# port: 1234

# Set the proxy scheme (e.g., `http` or
# `https`). It must be set with host and port.
# scheme: http

# Set this property as an alternate for `scheme://host:port`. It takes
# precedence over the other settings, if specified; however, an error
# will be thrown if both the URL and individual properties are set.
# url: NEEDS_TO_BE_SET

# Set the proxy user.
# user: NEEDS_TO_BE_SET

# Set the proxy password.
# pass: NEEDS_TO_BE_SET

# Set the proxy authentication type. Value
# options are `NTLM`, `Digest`, and `Basic`.
# auth_type: NEEDS_TO_BE_SET

#
=====
==
# agent
# Use the properties in this section to control the way and frequency
# with which the agent communicates to logs and the Contrast UI.
#
```



```
=====
==
# agent:

#
=====
# agent.diagnostics
# Use the properties in this section to specify the information the agent
# should collect and report in order to diagnose problems in the agent.
#
#
=====
# diagnostics:

# Creates config and system info files
# at startup if true. True by default.
#
# The same thing can be achieved by setting the
# CONTRAST__AGENT__DIAGNOSTICS__ENABLE=[true/false] env variable.
#
# enable: true

#
=====
# agent.route_coverage
# Use the following properties for the route-based coverage feature.
#
=====
# route_coverage: {}

#
=====
# agent.reporting
# Use the following settings to configure reporting to the Contrast UI.
#
=====
# reporting:

# Set the grace period (in milliseconds) after
# agent shutdown to allow draining pending reports.
# shutdown_grace_period_ms: 120000

#
=====
# agent.effective_config
# None
#
=====
# effective_config:

#
=====
# agent.effective_config.reporting
# None
#
```

```
=====
# reporting:

# Defaults to `true`. Controls whether configuration
# setting reports are sent to the Contrast web interface.
# enable: true

#
=====
# agent.logger
# Define the following properties to set logging values.
# If the following properties are not defined, the
# agent uses the logging values from the Contrast UI.
#
=====
# logger:

# Enable diagnostic logging by setting a path to a log file.
# While diagnostic logging hurts performance, it generates
# useful information for debugging Contrast. The value set here
# is the location to which the agent saves log output. If no
# log file exists at this location, the agent creates a file.
#
# Example - `/opt/Contrast/contrast.log` creates a log in the
# `/opt/Contrast` directory, and rotates it automatically as needed.
#
# path: ./contrast_agent.log

# Set the the log output level. Valid options are
# `ERROR`, `WARN`, `INFO`, `DEBUG`, and `TRACE`.
# level: INFO

# Set to `true` to redirect all logs to
# `stdout` instead of the file system.
# stdout: false

# Set to `true` to redirect all logs to `stderr` instead of
# the file system. May be combined with the corresponding
# `stdout` configuration to write to both streams.
# stderr: false

# Change the Contrast logger from a file-sized based rolling scheme
# to a date-based rolling scheme. At midnight server time, the
# previous day log is renamed to *file_name.yyyy-MM-dd*. Note -
# this scheme does not have a size limit; manual log pruning is
# required. You must set this flag to use the backups and size flags.
# roll_daily: false

# Set the roll size for log files in megabytes. The agent will
# attempt to prevent the log file from being larger than this size.
# roll_size: 100

# Set the number of backup files to keep. Set to `0` to disable.
# backups: 10
```

```
#
=====
# agent.security_logger
# Define the following properties to set security logging
# values associated with Protect. If not defined, the agent
# uses the security logging (CEF) values from the Contrast UI.
#
=====
# security_logger:

# Set the file to which the agent logs security events.
# path: ./contrast/security.log

# Set the log level for security logging. Valid options
# are `ERROR`, `WARN`, `INFO`, `DEBUG`, and `TRACE`.
# level: ERROR

# Change the Contrast security logger from a file-sized based rolling
# scheme to a date-based rolling scheme. At midnight server time,
# the log from the previous day is renamed to *file_name.yyyy-MM-dd*.
# Note - this scheme does not have a size limit; manual log
# pruning will be required. This flag must be set to use the
# backups and size flags. Value options are `true` or `false`.
# roll_daily: NEEDS_TO_BE_SET

# Specify the file size cap (in MB) of each log file.
# roll_size: NEEDS_TO_BE_SET

# Specify the number of backup logs that the agent will create before
# Contrast cleans up the oldest file. A value of `0` means that no
backups
# are created, and the log is truncated when it reaches its size cap.
#
# Note - this property must be used with
# `agent.security_logger.roll_daily=false`; otherwise,
# Contrast continues to log daily and disregard this limit.
#
# backups: NEEDS_TO_BE_SET

#
=====
# agent.security_logger.syslog
# Define the following properties to set Syslog values. If the
properties
# are not defined, the agent uses the Syslog values from the Contrast
UI.
#
=====
# syslog:

# Set to `true` to enable Syslog logging.
# enable: NEEDS_TO_BE_SET

# Set the IP address of the Syslog server
# to which the agent should send messages.
```

```
# ip: NEEDS_TO_BE_SET

# Set the port of the Syslog server to
# which the agent should send messages.
# port: NEEDS_TO_BE_SET

# Set the facility code of the messages the agent sends to Syslog.
# facility: 19

# Set the log level of Exploited attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_exploited: ALERT

# Set the log level of Blocked attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_blocked: NOTICE

# Set the log level of Blocked At Perimeter
# attacks. Value options are `ALERT`, `CRITICAL`,
# `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_blocked_perimeter: NOTICE

# Set the log level of Probed attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_probed: WARNING

# Set the log level of Suspicious attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_suspicious: WARNING

#
=====
# agent.security_logger.syslog.heartbeat
# Define the following properties to
# set the Syslog heartbeat properties.
#
=====
# heartbeat:

# Set to `true` to enable the Syslog heartbeat.
# The heartbeat will issue a Syslog message at
# the INFO level after every interval passes.
# enable: false

# Set the interval for sending heartbeat messages
# to the Syslog server (in milliseconds).
# interval_ms: 60000

#
=====
# agent.java
# The following properties apply to any Java agent-wide configurations.
#
=====
# java:
```

```
# Configure the Java agent to skip its application discovery
# algorithm, and instead associate all libraries, vulnerabilities,
# and web traffic to a single application with the name specified
# by this property. This configuration is preferred when deploying
# Java SE applications with embedded web servers (e.g., applications
# built with Spring Boot, Dropwizard, and embedded Jetty). When used
# with an application server, this configuration associates all
# web traffic with the single, standalone application, including
# web traffic handled by application server-hosted endpoints that
# would not be associated with a discovered application otherwise.
#
# Note - This settings takes preferences
# over the `application.name` setting.
#
# standalone_app_name: NEEDS_TO_BE_SET

# By default, the Java agent visits all classes at startup to look
# for vulnerabilities, which the agent may detect by scanning a
# class (e.g., hardcoded passwords). Set this property to `false`
# to disable the default behavior. If disabled, the agent will
# only visit classes which are likely to require sensors; this
# can improve application startup time, but may produce fewer
# findings (most likely findings that require static analysis).
#
# scan_all_classes: true

# By default, the Java agent deeply inspects all JAR and WAR files
loaded
# by the JVM to build a comprehensive understanding of the type
hierarchy.
# This understanding allows Contrast to instrument sensors into types
# that it might have overlooked. In most cases, this produces a slight
# increase in accuracy at the cost of increased application startup
# time. Set this property to `false` to disable this level of
inspection.
#
# scan_all_code_sources: true

#
=====
==
# inventory
# Use the properties in this section to override the inventory features.
#
=====
==
# inventory:

# Set to `false` to disable inventory features in the agent.
# enable: true

# Define a list of directories where libraries are stored.
# Directories must be formatted as a semicolon-delimited
# list for Windows or a colon-delimited list for Linux.
```

```
#
# Define a list of directories where libraries are stored.
# Directories must be formatted as a semicolon-delimited list.
# Example - `path1;path2;path3` (Windows) or `path1:path2:path3` (Linux)
# Example - `path1;path2;path3`
#
# library_dirs: NEEDS_TO_BE_SET

# Set the maximum archive unpacking depth when analyzing libraries.
# library_depth: 10

# Set the boolean to more aggressively limit the
# manifest information reported for libraries. If true,
# the limit is 1,000 characters, otherwise it's 3,000.
# prune_package_details: true

# Apply a list of labels to libraries. Labels
# must be formatted as a comma-delimited list.
# Example - `label1, label2, label3`
#
# tags: NEEDS_TO_BE_SET

#
=====
==
# assess
# Use the properties in this section to control Assess.
#
=====
==
# assess:

# Include this property to determine if the Assess
# feature should be enabled. If this property is not
# present, the decision is delegated to the Contrast UI.
# enable: false

# Control the values captured by Assess vulnerability events. `Full`
# captures most values by calling ToString on objects, which can
# provide more info but causes increased memory usage. `Minimal`
# has better performance as it only captures String type objects
# as strings and uses type name for other object type values.
# event_detail: minimal

# Apply a list of labels to vulnerabilities and preflight
# messages. Labels must be formatted as a comma-delimited list.
# Example - `label1, label2, label3`
#
# tags: NEEDS_TO_BE_SET

#
=====
# assess.sampling
# Use the following properties to control sampling in the agent.
#
```

```
=====
# sampling:

# Set to `true` to enable sampling.
# enable: false

# This property indicates the number of requests
# to analyze in each window before sampling begins.
# baseline: 5

# This property indicates that every *nth*
# request after the baseline is analyzed.
# request_frequency: 10

# This property indicates the duration for which a sample set is valid.
# window_ms: 180_000

#
=====
# assess.rules
# Use the following properties to control simple rule configurations.
#
=====
# rules:

# Define a list of Assess rules to disable in the agent. To view a
# list of rule names, in Contrast go to user menu > Policy Management >
# Assess rules. The rules must be formatted as a comma-delimited list.
#
# Example - Set `reflected-xss,sql-injection` to disable
# the reflected-xss rule and the sql-injection rule.
#
# disabled_rules: NEEDS_TO_BE_SET

#
=====
==
# profile
# Set configuration values under a profile name to enable
# multi-tenant application configuration on web servers. See
# https://support.contrastsecurity.com/hc/en-us/articles/360052187171-Multi-Application-configuration-with-Contrast-Profiles
# for more details.
#
=====
==
# profile: {}

#
=====
==
# protect
# Use the properties in this section to override Protect features.
#
=====
```

```
==
# protect:

# Include this property to determine if the Protect
# feature should be enabled. If this property is not
# present, the decision is delegated to the Contrast UI.
# enable: false

#
=====
# protect.rules
# Use the following properties to set simple rule configurations.
#
=====
# rules:

# Define a list of Protect rules to disable in the agent. To view a
# list of rule names, in Contrast go to user menu > Policy Management >
# Protect rules. The rules must be formatted as a comma-delimited list.
# disabled_rules: NEEDS_TO_BE_SET

#
=====
# protect.rules.bot-blocker
# Use the following selection to configure if the
# agent blocks bots. Set to `true` to enable blocking.
#
=====
# bot-blocker:

# Set to `true` for the agent to block known bots.
# enable: false

#
=====
# protect.rules.sql-injection
# Use the following settings to configure the sql-injection rule.
#
=====
# sql-injection:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or off.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

# Tell the agent to detect when semantic analysis of the query
# reveals tautologies used in exfiltration attacks (e.g., "or
# 1=1" or "or 2<>3"). The agent blocks if blocking is enabled.
# detect_tautologies: false

# Tell the agent to detect when semantic analysis of the query
```



```
# reveals the invocation of dangerous functions typically used in
# weaponized exploits. The agent blocks if blocking is enabled.
# detect_dangerous_functions: false

# Tell the agent to detect when semantic analysis of the query
# reveals chained queries, which is uncommon in normal usage but
# common in exploit. The agent blocks if blocking is enabled.
# detect_chained_queries: false

# Tell the agent to detect when semantic analysis of the query
# reveals database queries are being made for system tables and
# sensitive information. The agent blocks if blocking is enabled.
# detect_suspicious_unions: false

# Tell the agent to be more aggressive in detecting user
# inputs as SQL comments. This enables the agent to better
# detect SQL Injection input vectors that use comments to
# terminate queries. The agent blocks if blocking is enabled.
# aggressive_comment: false

#
=====
# protect.rules.cmd-injection
# Use the following properties to configure
# how the command injection rule works.
#
=====
# cmd-injection:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

# Detect when the agent sees user parameters being executed as
# system commands. The agent blocks if blocking is enabled.
# detect_parameter_command_backdoors: true

# Detect when a system command is issued which contains
# chained commands. The agent blocks if blocking is enabled.
# detect_chained_commands: true

# Detect when a system command is issued with an argument matching a
# known dangerous file path. The agent blocks if blocking is enabled.
# detect_dangerous_path_args: true

# Tell the agent to detect when commands come directly
# from input. The agent blocks if blocking is enabled.
# detect_phased_commands: true

#
=====
```

```
# protect.rules.cmd-injection-process-hardening
# Use the following settings to configure whether
# the agent blocks all attempts to start an external
# process. To enable blocking, set to 'true'.
#
=====
# cmd-injection-process-hardening:

# Set to `true` to enable the agent to block
# all attempts to start external processes.
# enable: false

#
=====
# protect.rules.path-traversal
# Use the following properties to configure
# how the path traversal rule works.
#
=====
# path-traversal:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

# Detect when custom code attempts to access sensitive
# system files. The agent blocks if blocking is enabled.
# detect_custom_code_accessing_system_files: true

# Detect when users attempt to bypass filters by
# using ":::$DATA" channels or null bytes in file
# names. The agent blocks if blocking is enabled.
# detect_common_file_exploits: true

#
=====
# protect.rules.method-tampering
# Use the following properties to configure
# how the method tampering rule works.
#
=====
# method-tampering:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off
```

```
#
=====
# protect.rules.reflected-xss
# Use the following properties to configure how
# the reflected cross-site scripting rule works.
#
=====
# reflected-xss:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.xxe
# Use the following properties to configure
# how the XML external entity works.
#
=====
# xxe:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.padding-oracle
# Use the following properties to configure
# how the padding-oracle rule works.
#
=====
# padding-oracle: {}

#
=====
==
# application
# Use the properties in this section for
# the application(s) hosting this agent.
#
=====
==
# application:

# Override the reported application name.
```

```
#
# Note - On systems where multiple, distinct applications may be served
# by a single process, this configuration causes the agent to report
# all discovered applications as one application with the given name.
#
# Note - On Java systems where multiple, distinct applications may be
# served by a single process, this configuration causes the agent to
report
# all discovered applications as one application with the given name.
#
# name: NEEDS_TO_BE_SET

# Override the reported application path.
# path: NEEDS_TO_BE_SET

# Add the name of the application group with which this
# application should be associated in the Contrast UI.
# group: NEEDS_TO_BE_SET

# Add the application code this application should use in the Contrast UI.
# code: NEEDS_TO_BE_SET

# Override the reported application version.
# version: NEEDS_TO_BE_SET

# Apply labels to an application. Labels must
# be formatted as a comma-delimited list.
# Example - `label1,label2,label3`
#
# tags: NEEDS_TO_BE_SET

# Define a set of `key=value` pairs (which conforms to RFC 2253) for
# specifying user-defined metadata associated with the application. The
# set must be formatted as a comma-delimited list of `key=value` pairs.
# Example - `business-unit=accounting, office=Baltimore`
#
# metadata: NEEDS_TO_BE_SET

# Provide the ID of a session which already exists in the Contrast
# UI. Vulnerabilities discovered by the agent are associated with
# this session. If an invalid ID is supplied, the agent will be
# disabled. This option and `application.session_metadata` are
# mutually exclusive; if both are set, the agent will be disabled.
# session_id: NEEDS_TO_BE_SET

# Provide metadata which is used to create a new session ID in the
# Contrast UI. Vulnerabilities discovered by the agent are associated with
# this new session. This value should be formatted as `key=value` pairs
# (conforming to RFC 2253). Available key names for this configuration
# are branchName, buildNumber, commitHash, committer, gitTag, repository,
# testRun, and version. This option and `application.session_id` are
# mutually exclusive; if both are set the agent will be disabled.
# session_metadata: NEEDS_TO_BE_SET

#
```

```
=====
==
# server
# Use the settings in this section to set metadata for the server
# hosting this agent. Contrast recognizes common, supported server
# names, paths, types and environments. Doing this may require a new
# server or license, and it may affect functionality of some features.
#
=====
==
# server:

# Override the reported server name.
# name: localhost

# Override the reported server path.
# path: NEEDS_TO_BE_SET

# Override the reported server type.
# type: NEEDS_TO_BE_SET

# Set the environment directly to override the default set
# by the Contrast UI. This allows the user to configure the
# environment dynamically at startup rather than manually
# updating the Server in the Contrast UI themselves afterwards.
#
# Valid values include `QA`, `PRODUCTION` and `DEVELOPMENT`.
# For example, `PRODUCTION` registers this Server as
# running in a `PRODUCTION` environment, regardless of the
# organization's default environment in the Contrast UI.
#
# environment: NEEDS_TO_BE_SET

# Apply a list of labels to the server. Labels
# must be formatted as a comma-delimited list.
# Example - `label1,label2,label3`
#
# tags: NEEDS_TO_BE_SET

# Set to `false` to disable detection of cloud
# provider metadata such as resource identifiers.
# discover_cloud_resource: true
```

Configure the Java agent for standalone applications



NOTE

Standalone application configuration is not needed with the Java agent 4.X.

Transport Layer Security (TLS)

The Contrast Java agent uses a secure TLS connection to communicate with Contrast.

For hosted customers, Contrast uses strong TLSv1.2 connections and certificates signed by industry standard certificate authorities (CAs). However, on-premises customers may need to configure the Java agent to use enterprise CAs, and may want the Java agent to send client certificates in the TLS handshake.

The Contrast Java agent uses the standard [Java Cryptography Architecture](#) for configuring TLS. Specifically, the Java agent uses the system's "TLS" [javax.net.ssl.SSLContext](#). For most users, this means that you can adjust the certificates trusted by the agent using the standard `javax.net.ssl.trustStore` system properties. You can also adjust the certificate the agent sends when the TLS server requests a client certificate using the standard `javax.net.ssl.keyStore` system properties.

This example configures the Java agent to use a custom key store and trust store:

```
java \
  -javaagent:contrast.jar \
  -Djavax.net.ssl.trustStore=/etc/pki/tls/my-enterprise-truststore.p12 \
  -Djavax.net.ssl.trustStorePassword=changeit \
  -Djavax.net.ssl.trustStoreType=PKCS12 \
  -Djavax.net.ssl.keyStore=/etc/pki/tls/server-client-certificate.p12 \
  -Djavax.net.ssl.keyStorePassword=password \
  -Djavax.net.ssl.keyStoreType=PKCS12 \
  -jar my-server.jar
```

Use the Java Agent with the Java Platform Module System (JPMS)

JPMS is a way to encapsulate code that is present in Java versions 9 and later. Contrast supports inspection of modules and launching of applications written with the JPMS.

The Java Agent requires that the `java.sql` package be required by the application's `module-info.java` files:

```
module mymodule {
    requires java.sql;
}
```

or supplied by the `--add-modules` command-line argument at runtime:

```
java -javaagent:/opt/contrast/contrast-agent.jar --add-modules java.sql --
module-path libs --module mymodule/mycompany.App
```

Java 2 security

The Java 2 security manager allows system administrators to enforce policies that dictate the permissions available to Java code within a JVM.

If you are using the Java 2 security manager with the Java agent, you will need to configure Java security policy files to apply permissions to Java code principals.

Java code principals are typically identified by a `CodeSource` (like, a JAR), and in rare cases, by the entity that signed the JAR.

For example, in Tomcat's [default catalina.policy file](#), the policy grants permissions to the JDBC driver JAR:

```
// The permission granted to your JDBC driver
grant codeBase "jar:file:${catalina.base}/webapps/examples/WEB-INF/lib/
driver.jar!/-" {
    permission java.net.SocketPermission "dbhost.mycompany.com:5432",
```

```
"connect" ;  
};
```

The Java 2 security manager can be useful in situations where the system administrator can't fully trust the code deployed by users. For example, if you are hosting users' applications on multi-tenant Tomcat instances, you could use the Java 2 security manager to constrain users' applications from taking down their whole service (for example, by disallowing calls to `System.exit()`).

If you are using the Java 2 security manager with the Contrast Java agent, you should grant the Java agent the full set of permissions in your security policy file (*java.security.AllPermission*). To do this, replace `<YourContrastJarPath>` with the path to your [Contrast JAR \(page 120\)](#), and use:

```
grant codeBase "file:<YourContrastJarPath>" {  
    permission java.security.AllPermission;  
};
```

If you are using Java 2 security manager and one of these environments, you may also need to complete further configuration:

- [Glassfish](#)
- [Jetty \(page 144\)](#)
- [Tomcat \(page 145\)](#)
- [WebLogic \(page 146\)](#)
- [WebSphere \(page 147\)](#)
- [WildFly \(page 143\)](#)

Java agent telemetry

The Contrast Java agent use telemetry to collect usage data. Telemetry is collected when an instrumented application first loads the agent's sensors and then periodically (every few hours) afterwards.

[Your privacy is important to us \(page 1275\)](#). The telemetry feature doesn't collect application data. The data is anonymized before being sent securely to Contrast. Then the aggregated data is stored encrypted and under restricted access control. Any collected data will be deleted after one year.

The telemetry feature collects the following data (optional):

Agent version	Data collected
Java 3.16.XXXX	<ul style="list-style-type: none">• Operating system and version• Whether the agent is running in a container• Memory limits configured in the JVM• Java version and vendor• Physical memory available• CPU count



NOTE

To opt-out of the telemetry feature, set `CONTRAST_AGENT_TELEMETRY_OPTOUT` environment variable to `true` or `1`. Telemetry data is securely sent to `telemetry.java.contrastsecurity.com`. You can also opt out of telemetry by blocking communication at the network level.

.NET Framework agent

The Contrast .NET Framework agent analyzes the behavior of .NET web applications as users interact with these applications.



NOTE

The latest .NET Framework agent supports Assess (IAST), Protect (RASP), and SCA features.

Once installed, the .NET Framework agent automatically instruments ASP.NET applications deployed to IIS. Agent analysis is performed as applications are exercised by users (or by automated scripts or tests).

You can view the results of the agent's analysis in the Contrast application. The Contrast .NET Framework agent consists of several components:

- **Background Windows service:** (*DotnetAgentService.exe*) This service prepares the environment for instrumentation and manages communication between agent components. This is the main service that controls agent behavior. You can disable Contrast's instrumentation and analysis by stopping the agent's background Windows service.
- **The .NET Profiler:** This instruments applications to weave in method calls out to agent sensors.
- **Sensors:** These gather security, architecture and library information.
- **The .NET Agent Explorer (page 266):** This is a Windows system tray application that displays high-level information about the health of the agent.

As a next step, you can:

- [Install the .NET Framework agent \(page 214\)](#)
- [View supported technologies \(page 212\)](#)
- [View system requirements \(page 213\)](#)
- [Use application pools in IIS \(page 268\)](#)

Supported technologies for the .NET Framework agent

The Contrast .NET agent supports analysis of web applications built on the following technologies.

Technology	Supported versions	Notes
.NET Framework for Windows		
Application runtime version	4.5 and later	<p>Most users are able to use the modern .NET Framework agent, even if their application targets an older version of .NET 4, due to .NET framework application compatibility.</p> <p>Not supported:</p> <ul style="list-style-type: none">• Classic ASP Classic ASP applications don't run on the .NET runtime.• Mono runtime The agent uses the CLR Profiling API to instrument applications. The CLR Profiling API is a Component Object Model (COM)-based interface exposed by the CLR. Linux does not support COM. Therefore Mono does not support the CLR Profiling API and Contrast cannot support Mono.
Server runtime version	4.7.1, 4.7.2, 4.8	
CLR	CLR4	

Technology	Supported versions	Notes
Web servers	<ul style="list-style-type: none">IISIIS Express	
Application frameworks	<ul style="list-style-type: none">ASP.NET MVC 3-5ASP.NET Web FormsASP.NET Web PagesIIS-Hosted ASMX-based Web ServicesIIS-Hosted Web APIIIS-Hosted WCF ServicesOWIN Hosted Web API (via a Windows service or a command line application)	<p>These frameworks are explicitly tested, however, you may still be able to analyze other applications if the framework simply wraps the typical ASP.NET classes (for example, System.Web.HttpRequest).</p> <p>Not supported:</p> <ul style="list-style-type: none">Analysis of .NET Framework ASP.NET Core applications (use our .NET Core agent (page 270) to analyze .NET Core applications).Applications running under partial trust.



NOTE

- For Azure App Service, .NET Framework applications must use the .NET Framework [site extension](#) or [NuGet package](#).
- .NET Core applications must use the .NET Core-specific [site extension](#) or [NuGet package](#).

System requirements for .NET Framework agent

Before installing the .NET Framework agent, you must meet the following requirements:

- You have administrative access to a web server, and the server is supported by Contrast.
- There is a deployed application to be analyzed, and the web application technology is supported by Contrast.
- IIS can be restarted.
- The web server has network connectivity with Contrast.
- The server meets the minimum requirements.

Requirements	Recommended	Notes
Server runtime version	4.7.1 or later	While .NET 4.7.1 or later is required to install the .NET Framework agent, the agent can analyze applications that target .NET 4.5 and later due to .NET Framework application compatibility.
Operating system	<ul style="list-style-type: none">Windows 10Windows Server 2012, 2012 R2, 2016, 2019, 2022Azure Virtual Machines, Cloud Services, Mobile ServicesAzure App Service	
Processor architecture	<ul style="list-style-type: none">32-bit64-bit	On 64-bit systems, you can use the agent to analyze both 32-bit and 64-bit web applications.
CPU	At least 4	Minimum: 2

Requirements	Recommended	Notes
Memory	At least 8 GB The .NET agent roughly doubles the memory requirements of analyzed applications. Applications should use less than half of the available memory when the .NET agent is not installed.	Minimum: 4 GB
Server	<ul style="list-style-type: none">.NET Framework 4.7.1CLR 4 (.NET 4.0 and later).	For servers on earlier versions use the Legacy .Net Framework agent .



NOTE

- The .NET Framework agent uses the CLR Profiling API to perform data and code flow analysis (for example, detect SQL-injection, XSS, weak cryptography) as well as to detect libraries and technologies used by analyzed applications.
- The Contrast agent can [exist alongside other .NET Profiler agents \(page 264\)](#), such as performance or APM tools with the agent `.dotnet.enable_chaining` configuration setting enabled.
- For servers on earlier versions use the [Legacy .NET Framework agent](#).

Install the .NET Framework agent

In most deployments, much of the installation is done automatically by the installer or site extension. A basic installation of the .NET Framework agent looks like this:

- Download the installer and place it on the server.
- Run the installer.
- Use the application as you normally would and verify that Contrast sees your application.



NOTE

If you are using Windows 2008, a version of .NET Framework prior to 4.7.1 or if your application targets CLR2, you should use the [Legacy .Net Framework agent installer](#).

Specifically, installation varies depending on how you want to install the .NET Framework agent:

- [.NET Framework Windows installer \(page 214\)](#)
- [Azure App Service \(page 217\)](#)
- [Into a container, like Docker \(page 220\)](#)
- [Web API-OWIN \(page 223\)](#)

To auto-upgrade your agent, enable this option with the [Agent Upgrade Service \(page 226\)](#).

.NET Framework agent installer for Windows

The Contrast .NET Framework agent installer is a normal Windows application installer built using standard MSI technology. It validates that the target server satisfies several requirements (for example, that the server's operating system is a supported operating system). If all requirements are met, the installer:

- Registers the .NET Framework agent as a standard Windows program.
- Places the agent's files on a disk in the specified install location (for example, *C:\Program Files\Contrast\dotnet*). This includes several dynamic link libraries (DLLs) and executables, such as the background Windows service that drives agent behavior.
- Creates the specified data directory for the agent that's primarily used to store agent log files and configuration (for example, *C:\ProgramData\Contrast\dotnet*).
- Registers the agent's background Windows service with the operating system.
- Adds the agent's Native Module to IIS. The Native Module registers the agent's profiler component with IIS through environment variables. This causes the CLR to load the agent's profiler, which is responsible for instrumenting analyzed applications.
- Starts the agent's background Windows service and [.NET Agent Explorer \(page 266\)](#) application. The background service is responsible for:
 - Communication with profiler and sensor components through local named pipes.



NOTE

- If you are using the agent with [self-hosted Web API and OWIN \(page 223\)](#) (outside of IIS), further configuration is needed.
- If you are using Windows 2008, a version of .NET Framework prior to 4.7.1 or if your application targets CLR2, you should use the [Legacy .NET Framework agent installer](#).

Install the .NET Framework agent using Contrast

1. In the Contrast web application, select **Add new** in the top right.
2. Select the **Application** card.
3. Choose **.NET Framework** in the application language dropdown, then select **IIS hosted** and select the link to **Download the agent and YAML configuration file**.
4. Extract the downloaded ZIP archive on the web server, and run *ContrastSetup.exe*. This installs the .NET Framework agent.

The *contrast_security.yaml* file is copied to the agent's data directory by the installer and placed in *C:\ProgramData\Contrast\dotnet\contrast_security.yaml* by default. The installer does not copy the YAML file if it already exists at the destination.

 - You can [use the command line \(page 228\)](#) to access additional options supported by the .NET Framework agent installer for Windows.
 - If you are using another profiler in this environment, such as an APM like New Relic or AppDynamics, then you need to enable [Contrast profiler chaining \(page 264\)](#).
5. You can further configure the agent using the [.NET Framework YAML template \(page 230\)](#).
6. Use the application as you normally would and verify that Contrast sees your application.

If there are some applications you don't need to analyze, or if you are trying to be lean on performance, consider using [application pools \(page 268\)](#) to limit the number of applications instrumented.

Install the .NET Framework agent using command line

You can use the command line to access additional options supported by the .NET Framework agent installer for Windows.

The .NET agent can be installed using the Windows UI, and uninstalled or repaired using standard Windows features (including the Programs and Features Control Panel and Powershell). However, you

may want to use the .NET Framework agent installer for Windows to perform these actions instead for certain scenarios such as automated scripting.


Use these commands for attended mode:

- **Install:** ContrastSetup.exe
- **Uninstall:** ContrastSetup.exe -uninstall
- **Repair:** ContrastSetup.exe -repair

Use these commands for unattended or silent mode:

- **Install:** ContrastSetup.exe -s -norestart
- **Uninstall:** ContrastSetup.exe -uninstall -s -norestart
- **Repair:** ContrastSetup.exe -repair -s -norestart

The .NET Framework agent installer for Windows supports several additional options that are accessible when you use the command line for installation.

Option	Description	Example
INSTALLFOLDER	This option specifies the install directory. Program files will be written to this directory. Defaults are dependent on OS variables.	INSTALLFOLDER="D:\Programs\Contrast"
AGENT_EXPLORER_INSTALLFOLDER	This option specifies the directory for Agent Explorer files.	AGENT_EXPLORER_INSTALLFOLDER="C:\Program Files\Contrast\agent-explorer"
DATAFOLDER	This option specifies the data directory. Logs and the contrast_security.yaml file will be written to this directory. Defaults are dependent on OS variables.	DATAFOLDER = "D:\Data\Contrast"
PathToYaml	This option specifies a custom YAML configuration file. The default value is the <i>contrast_security.yaml</i> file located relative to the installer's location.	PathToYaml=c:\contrast_security.yaml
SERVICE_STARTUP_TYPE_MANUAL	This option is required when you install, upgrade, and repair the agent. If you set the value to 1, this option sets the Contrast service startup type to Manual. The default value is 0 (Automatic Delayed Start).	SERVICE_STARTUP_TYPE_MANUAL=1
SUPPRESS_SERVICE_START	This option is required when you install, upgrade, and repair the agent. If you set the value to 1, this option automatically suppresses starting the service. The default value is 0.	SUPPRESS_SERVICE_START=1
SUPPRESS_RESTARTING_IIS	If you set the value of this option to 1, the installer does not restart IIS. The default value is 0.	SUPPRESS_RESTARTING_IIS=0
<div>  <div> <p>NOTE</p> <ul style="list-style-type: none"> • Applications do not load the agent until IIS restarts. • Setting SUPPRESS_RESTARTING_IIS will prevent an auto-upgrade from running unless IIS does not have any active workers when the upgrade runs. </div> </div>		
USE_VIRTUAL_SERVICE_ACCOUNT	Run the agent service under a restricted virtual service account. Configures the service to run as the NT Service\DotnetAgentSvc virtual account instead of SYSTEM.	USE_VIRTUAL_SERVICE_ACCOUNT=0
INSECURE_YAML_FILE	Restrict edits to the contrast_security.yaml file of non-elevated users.	INSECURE_YAML_FILE=0

Option	Description	Example
INSTALL_AGENT_EXPLORER	If you don't want to install the Agent Explorer, set the value for this option to 0. The default value is 1, which installs the Agent Explorer.	INSTALL_AGENT_EXPLORER=1
INSTALL_UPGRADE_SERVICE	If you don't want to install the agent upgrade service, set the value of this option to 0. The default value is 1 which installs the agent upgrade service.	INSTALL_UPGRADE_SERVICE=1
UPGRADE_SERVICE_INSTALLFOLDER	This option specifies the directory for the upgrade service files.	UPGRADE_SERVICE_INSTALLFOLDER="C:\Program Files (x86)\Contrast\upgrade-service"

**TIP**

To install the .NET agent using scripts, you can use this command:

```
ContrastSetup.exe -s PathToYaml=C:\Temp\custom.yaml
```

This command installs the .NET agent in silent and unattended mode and uses a custom path to the YAML configuration file.

**IMPORTANT**

Contrast automatically restarts IIS when you install the agent.

Install the .NET Framework agent for Azure App Service

Use this procedure for an express installation of the .NET Framework agent using Azure Portal Extensions.

Before you begin

Before you begin, check the [system requirements \(page 213\)](#) and [supported technologies \(page 212\)](#) to be sure installation will work and ensure best performance.

Steps

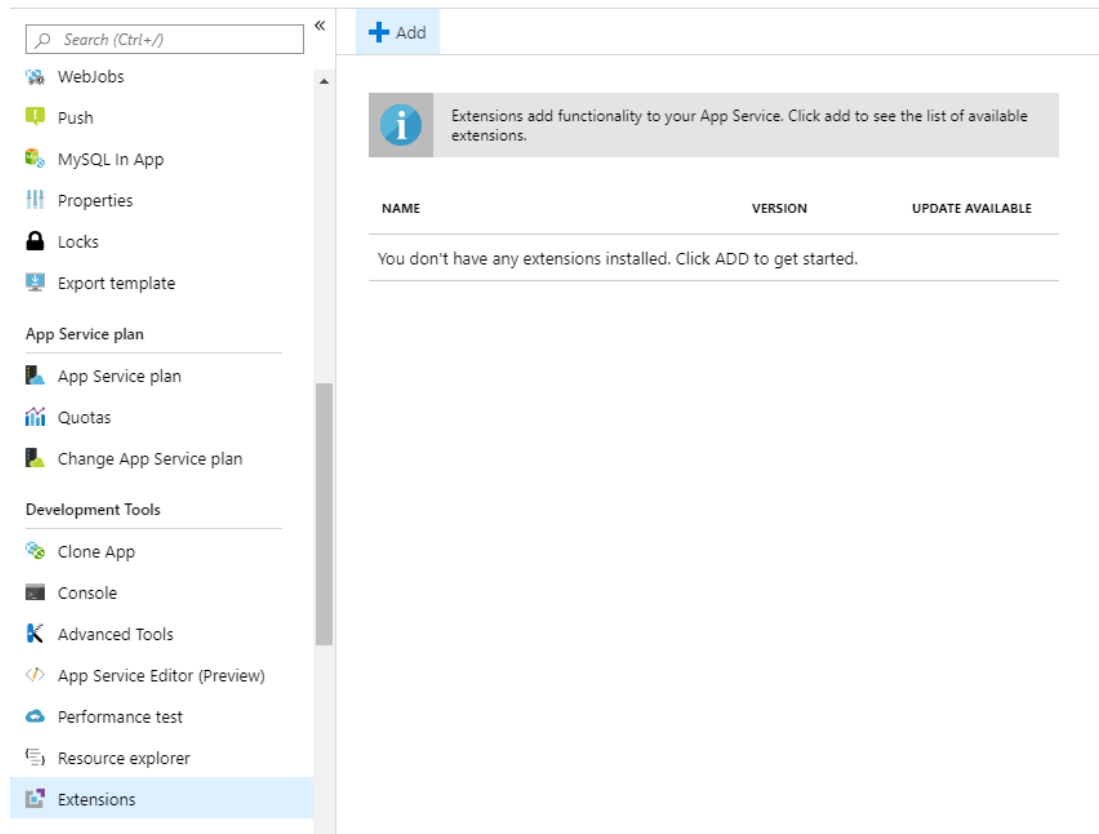
1. Create an [Azure account](#), if you don't have one already.
2. Create a [.NET web application](#) and deploy it to Azure App Service.
3. Publish your application to Azure, and confirm that it works as expected without Contrast.
4. Ensure that your application is deployed using a Windows plan. (Linux plans do not support Site Extensions.)

**NOTE**

If you do not have access to the site extension, you can [install the .NET Framework agent manually with NuGet \(page 223\)](#).

5. Add the Contrast .NET Framework Agent Site Extension:
 - With the Azure portal
 - a. In the Azure Portal, select your hosted application.

b. Select **Extensions**.



c. Select **Add**.

d. Select the **Contrast .NET Framework Site Extension for Azure App Service**. This is the extension for .NET Framework applications.

e. Select **OK**, and agree to the terms and conditions.

f. Wait a few seconds and confirm the site extension installed correctly.

**NOTE**

The site extension sets a number of environment variables, including:

```
COR_ENABLE_PROFILING=1
COR_PROFILER={EFEB8EE0-6D39-4347-A5FE-4D0C88BC5BC1}
COR_PROFILER_PATH_32=D:\home\siteextensions\Contrast.Net.Azure.SiteExtension\ContrastAppService\runtimes\win-x86\native\ContrastProfiler.dll
COR_PROFILER_PATH_64=D:\home\siteextensions\Contrast.Net.Azure.SiteExtension\ContrastAppService\runtimes\win-x64\native\ContrastProfiler.dll
CONTRAST_INSTALL_DIRECTORY=D:\home\siteextensions\Contrast.Net.Azure.SiteExtension\ContrastAppService\
MicrosoftInstrumentationEngine_ConfigPath32_ContrastX86Config=D:\home\siteextensions\Contrast.Net.Azure.SiteExtension\runtimes\win-x86\ContrastCieProfiler.config
MicrosoftInstrumentationEngine_ConfigPath64_ContrastX64Config=D:\home\siteextensions\Contrast.Net.Azure.SiteExtension\runtimes\win-x64\ContrastCieProfiler.config
```

If the CLR instrumentation engine (CIE) is configured for the application (for example, because Application Insights is enabled), Azure should automatically overwrite the `CORECLR_PROFILER*` variables to point to the profiler of the CIE.

The CIE will then use the `MicrosoftInstrumentationEngine_*` variables to load the Contrast agent.

If the CIE is not configured for the application, the standard `CORECLR_PROFILER*` variables will be used to load the Contrast agent.

- With the Azure CLI
 - Use a command similar to this one for a .NET Framework Site Extension

```
az resource create --resource-group 'myResourceGroup' --resource-type Microsoft.Web/sites/siteextensions --name myAppService/siteextensions/Contrast.NET.Azure.SiteExtension --properties "{}"
```

In this example, the command adds a Contrast .NET Framework Site Extension to an App Service named "myAppService" in Resource Group "myResourceGroup".

After you add the extension, the Azure Portal displays a list of the installed agents with details similar to the following:

Name	Version	Update Available
Contrast.NET Framework Site Extension for Azure App Service	51.0.22	No

**TIP**

You can also install the agent from the Site Extensions area of your application management SCM (Kudu) site.

**IMPORTANT**

If a new version of the .NET Framework agent is available, it's indicated in the Azure Portal or Kudu dashboard. You must stop the site before starting the update; otherwise, the update may fail.

6. Add configuration options

- With the Azure Portal
 - a. In the Azure Portal, select your hosted application.
 - b. Select **Configuration** under **Settings** to configure settings that allow the agent to connect to Contrast.
 - c. Select **New application setting** and add the following values for your application:

Key	Value
CONTRAST__API__USER_NAME	Replace with your agent username (page 104) .
CONTRAST__API__SERVICE_KEY	Replace with your agent service key (page 104) .
CONTRAST__API__API_KEY	Replace with your agent API key (page 104) .
CONTRAST__API__URL	Defaults to https://app.contrastsecurity.com . Replace with another URL, if you're using a Contrast application that's hosted elsewhere.

- With the Azure CLI
 - Enter a command similar to this one:

```
az webapp config appsettings set --resource-group 'myResourceGroup'
--name 'myAppService' --settings CONTRAST__API__URL=https://
app.contrastsecurity.com CONTRAST__API__API_KEY={Your API
KEY} CONTRAST__API__SERVICE_KEY={Your Service key}
CONTRAST__API__USER_NAME={Your agent user}
```

Get API values ([agent keys \(page 104\)](#)) from the Contrast web interface or by downloading a YAML file for the or .NET Framework agent.

- 7. In the Azure Portal, go to the application overview and **Restart** the application.**
Running the application automatically instruments any application that is running inside of the App Service. You should begin to see data in Contrast
- 8. Navigate to the application and confirm the application is reporting to Contrast.**
You can view log files to verify that Contrast is running:
 - a. In the Azure Portal, go to **Advanced Tools** for the app service.
 - b. Select **Go**.
 - c. In the Kudu Services window, select “Debug console” menu at the top and select “CMD”.
 - d. Select the `LogFiles` directory.
 - e. Select the `Contrast` directory.
 - f. Select the `dotnet` directory.
You will see an agent log named `<PID>_Profiler_<App Service Name>_<XXX>.log`.
 - g. Verify that there are no ERROR log entries.

Install the .NET Framework agent using a container

Before you begin

This topic provides general guidance for installing the Contrast .NET Framework agent in a containerized application, with Docker as an example.

You should have a basic understanding of how containers and related software work. You may need to adjust the instructions to meet your specific circumstances.

Step 1: Install the agent

In this example, the latest .NET Framework agent is copied. Check DockerHub for available tags.

```
FROM mcr.microsoft.com/dotnet/framework/sdk:4.8

# Hidden for brevity...

# Copy the required agent files from the official Contrast agent image.
COPY --from=contrast/agent-dotnet-framework:latest C:\Contrast C:\Contrast
```

Step 2: Configure the agent

Contrast agents accept configuration from multiple sources, with order of precedence documented in the [order of precedence \(page 106\)](#) section.

A mixed approach is recommended:

- Use a YAML file so that you can share a common configuration between multiple applications.
- Use environment variables for application-specific configuration values, to override values specified in a YAML file, or for sensitive keys that are injected during runtime.

YAML file configuration:

When you use a [YAML file to configure \(page 107\)](#) the agent, you can use the environment variable `CONTRAST_CONFIG_PATH` to indicate where the YAML file is located inside the container.

For example, given a YAML file called `contrast_security.yaml` that exists in the Docker build context:

```
agent:
  logger:
    path: /var/tmp
    level: WARN
```

You can use the `CONTRAST_CONFIG_PATH` environment variable to add the agent YAML file to the container image as follows:

```
FROM mcr.microsoft.com/dotnet/framework/sdk:4.8

# Hidden for brevity...

# Add the Contrast agent to the image.
COPY --from=contrast/agent-dotnet-framework:latest C:\Contrast C:\Contrast


# Copy the contrast_security.yaml file from Docker build context.
COPY ./contrast_security.yaml /contrast_security.yaml

# Finally configure the agent to use the YAML file previously copied.
ENV CONTRAST_CONFIG_PATH=/contrast_security.yaml
```

Environment variable configuration:

To set an application-specific configuration, use [environment variables. \(page 110\)](#) This table contains some common configuration options.

Title	Usage	Environment variable
Application name	Specify the application name reported to Contrast.	<code>CONTRAST__APPLICATION__NAME</code>

Title	Usage	Environment variable
Application group	Specify the application access group for this application during onboarding.	CONTRAST__APPLICATION__GROUP
<div>  <p>NOTE Create application access groups in Contrast before using this variable.</p> </div>		
Application tags	Add labels to an application.	CONTRAST__APPLICATION__TAGS
Server name	Specify the server name reported to Contrast.	CONTRAST__SERVER__NAME
Server environment	Specify in which environment the application is running. Valid values for this configuration are: Development, QA, and Production.	CONTRAST__SERVER__ENVIRONMENT
Server tag	Add labels to the server.	CONTRAST__SERVER__TAG

Step 3: Add profiler variables and authentication credentials

To enable instrumentation of your application, the .NET agent requires [additional environment variables \(page 110\)](#). The `COR_CLR` variables load the agent and the `CONTRAST_` variables are for agent authentication to the server.

Using the Dockerfile example in this topic:

```
FROM mcr.microsoft.com/dotnet/framework/sdk:4.8

COPY --from=contrast/agent-dotnet-framework:latest C:\Contrast C:\Contrast

ENV COR_ENABLE_PROFILING=1 \
    COR_PROFILER={EFEB8EE0-6D39-4347-A5FE-4D0C88BC5BC1} \
    COR_PROFILER_PATH_32=C:\Contrast\runtimes\win-x86\native\ContrastProfiler.dll \
    COR_PROFILER_PATH_64=C:\Contrast\runtimes\win-x64\native\ContrastProfiler.dll
```

Additionally, the [following environment variables \(page 104\)](#) are required for agent authentication to the server.

- **Agent Token:** This variable is a base64 encoded JSON object containing the url, api_key, service_key, and user_name configuration settings, allowing you to set them set in a single variable.

If your agent configuration refers to both the legacy settings and the agent token, (in environment variables or the YAML file), the legacy settings take precedence. To use just the agent token value, make sure you remove references to the legacy settings.

```
set CONTRAST__API__TOKEN=<token-value
```

- **Legacy settings:** If you are using an agent version earlier than 51.0.40, set these variables:

```
CONTRAST__API__URL=https://app.contrastsecurity.com/Contrast
CONTRAST__API__API_KEY={Your API KEY here}
CONTRAST__API__SERVICE_KEY={Your Service key here}
```

Get the API values ([agent keys \(page 104\)](#)) from the Contrast web interface or by [downloading a YAML file \(page 230\)](#) for the .NET Framework agent.

Examples

You can see examples of finished code in this [GitHub repository](#). In particular, these ASP.NET application use cases might be helpful:

- Default AppPool:
 - [Dockerfile](#)
 - [Entrypoint script](#)
- Custom AppPool:
 - [Dockerfile](#)
 - [Entrypoint script](#)

See also

Contrast Support Portal [Kubernetes and Contrast](#)

Contrast Support Portal [AWS Fargate and Contrast agents](#)

Install the .NET Framework agent manually with NuGet

In some instances, you may prefer to manually install the .NET Framework agent using NuGet. For example, this can be useful if you are unable to access the [Azure App Service site extension \(page 217\)](#) or if you prefer to include the .NET Framework agent as a dependency.

1. Add the Contrast NuGet package to your application.
In Visual Studio, under the application project in the Solution Explorer, right-click on **References** and select **Manage NuGet Packages**.
Search for the **Contrast.Net.Azure.AppService** package, select it and add it to your project.
Build your application. Confirm that Contrast assemblies (for example, `ContrastProfiler.dll`) are in a new `contrastsecurity` folder that's created in the application's root directory.
2. Add application authentication settings for Contrast.
You can either add the authentication settings through the App Service Settings window in Visual Studio's "Publish to Azure App Service", or directly through the Azure App Service Portal.
Set the Contrast authentication keys that the agent needs to connect to Contrast, and select **Save**.
You can [find your keys \(page 104\)](#) in your profile.
3. Follow the build process from the dotnet source code repository.
4. Go to the **Application settings** area of your application in the Azure Portal. Set the Contrast authentication keys that the agent needs to connect to Contrast, and select **Save**.
5. Using Visual Studio, publish the application to Azure.
Once the application has loaded, use the application and then open Contrast to verify that the server and application are active, and that any expected vulnerabilities appear.

Install the .NET Framework agent with Web API and Owin

The .NET agent supports analysis of Web API applications that are self-hosted with the Open Web Interface for .NET (OWIN). The Web Api can be deployed as a command line application and Windows service.



NOTE

Web API applications hosted in the IIS integrated pipeline using the SystemWeb HttpModule and those deployed with an OWIN Host are **not supported**.

1. Install the .NET agent with [.NET Framework agent installer for Windows \(page 214\)](#).

2. Set the environment variables, depending on how you deploy your Web API hosted by OWIN.
 - **Deployed as a command line application:** Set these environment variables before running the command line application that is being used to self-host OWIN:

Environment variable	Value
COR_ENABLE_PROFILING	1
COR_PROFILER	{EFEB8EE0-6D39-4347-A5FE-4D0C88BC5BC1}
COR_PROFILER_PATH_32	C:\Program Files\Contrast\dotnet\runtimes\win-x86\native\ContrastProfiler.dll
COR_PROFILER_PATH_64	C:\Program Files\Contrast\dotnet\runtimes\win-x64\native\ContrastProfiler.dll

**NOTE**

COR_PROFILER_PATH_32 / COR_PROFILER_PATH_64 must match the installation directory chosen during the install of the .NET Framework agent.

- **Deployed as a Windows service:**

Install the service that contains the Web API application. Note the name of the service. Under the service's registry key, create a REG_MULTI_SZ value called `Environment`. If there is already an `Environment` value, add the new values below the existing values. Set the required environment variables. Each environment variable key/value pair must be separated by a new line. Environment variables that are unique for each service can be set under that service's registry key. The service's registry key can be found at: `HKLM\SYSTEM\CurrentControlSet\Services\YourServiceName`.

Environment variable	Value
COR_ENABLE_PROFILING	1
COR_PROFILER	{EFEB8EE0-6D39-4347-A5FE-4D0C88BC5BC1}
COR_PROFILER_PATH_32	C:\Program Files\Contrast\dotnet\runtimes\win-x86\native\ContrastProfiler.dll
COR_PROFILER_PATH_64	C:\Program Files\Contrast\dotnet\runtimes\win-x64\native\ContrastProfiler.dll
CONTRAST_CONFIG_PATH	C:\ProgramData\contrast\dotnet\contrast_security.yaml

**NOTE**

COR_PROFILER_PATH_32 / COR_PROFILER_PATH_64 must match the installation directory chosen during the install of the .NET Framework agent.

3. Restart the service so that new values are loaded. This PowerShell script can be used to set the required environment variables:

```
param (  
    # Name of the service that it was given at installation.  
    [Parameter(Mandatory=$true)]  
    [string]  
    $ServiceName,  
  
    # Path to the 64-bit Contrast profiler DLL.  
    # Defaults to: "C:\Program Files\Contrast\dotnet\runtimes\win-x64\native\ContrastProfiler.dll"  
    [string]  
    $ProfilerPath64 = "C:\Program Files\Contrast\dotnet\runtimes\win-
```

```
x64\native\ContrastProfiler.dll",

    # Path to the 32-bit Contrast profiler DLL.
    # Defaults to: "C:\Program Files\Contrast\dotnet\runtimes\win-
x86\native\ContrastProfiler.dll"
    [string]
    $ProfilerPath32 = "C:\Program Files\Contrast\dotnet\runtimes\win-
x86\native\ContrastProfiler.dll",

    # Path to the Contrast agent configuration YAML file.
    # Defaults to:
"C:\ProgramData\contrast\dotnet\contrast_security.yaml"
    [string]
    $ConfigYamlPath =
"C:\ProgramData\contrast\dotnet\contrast_security.yaml"
)

if (-Not (Test-Path -Path $ProfilerPath64 -PathType Leaf)) {
    Write-Host "Cannot find 64-bit profiler DLL at path
`"$ProfilerPath64`"."
    exit 1
}

if (-Not (Test-Path -Path $ConfigYamlPath -PathType Leaf)) {
    Write-Host "Cannot find configuration YAML file at path
`"$ConfigYamlPath`"."
    exit 1
}

if (-Not (Test-Path -Path $ProfilerPath32 -PathType Leaf)) {
    Write-Host "Cannot find 32-bit profiler DLL at path
`"$ProfilerPath32`"."
    exit 1
}

# Check if there is a service with the specified name installed.
$service = Get-Service -Name $ServiceName -ErrorAction Ignore

if ($null -Eq $service) {
    Write-Host "The service `"$ServiceName`" was not found."
    exit 2
}

# Create value for multiline registry string.
$values = @(
    "COR_ENABLE_PROFILING=1",
    "COR_PROFILER={EFEB8EE0-6D39-4347-A5FE-4D0C88BC5BC1}",
    "COR_PROFILER_PATH_64=$ProfilerPath64",
    "COR_PROFILER_PATH_32=$ProfilerPath32",
    "CONTRAST_CONFIG_PATH=$ConfigYamlPath"
)

$registryKey = "HKLM:\SYSTEM\CurrentControlSet\Services\$ServiceName"

# Check if the Environment value already exists.
```

```
$environmentValue = Get-ItemProperty -Path $registryKey -Name
"Environment" -ErrorAction Ignore

if ($null -Ne $environmentValue) {
    # Add the Contrast environment variables to the existing variables.
    $existingValues =
[System.Collections.ArrayList]@($environmentValue.Environment)
    foreach ($item in $values) {
        $idx = $existingValues.Add($item)
    }
    $values = $existingValues
}

# Set the environment variables for the service.
Set-ItemProperty -Path $registryKey -Type MultiString -Name
"Environment" -Value $values

# Restart the service so it picks up the new environment variables.
Restart-Service -Name $serviceName
```

Agent upgrade service

The Agent Upgrade Service is a background Windows service that helps you keep the .NET Framework and .NET Core for IIS agents automatically updated to the most recent version on Windows. The Agent Upgrade Service is included with the .NET Framework Agent Installer and .NET Core Agent for IIS Installer; the agent installers install two products:

- the corresponding agent, and
- the Agent Upgrade Service.

By default, the Agent Upgrade Service checks for new agent versions released to NuGet when the service first starts up (when the Windows Server is restarted.) If a new agent version is found, the Upgrade Service will download the new agent version, verify the installer's signature, and then finally execute the installer.

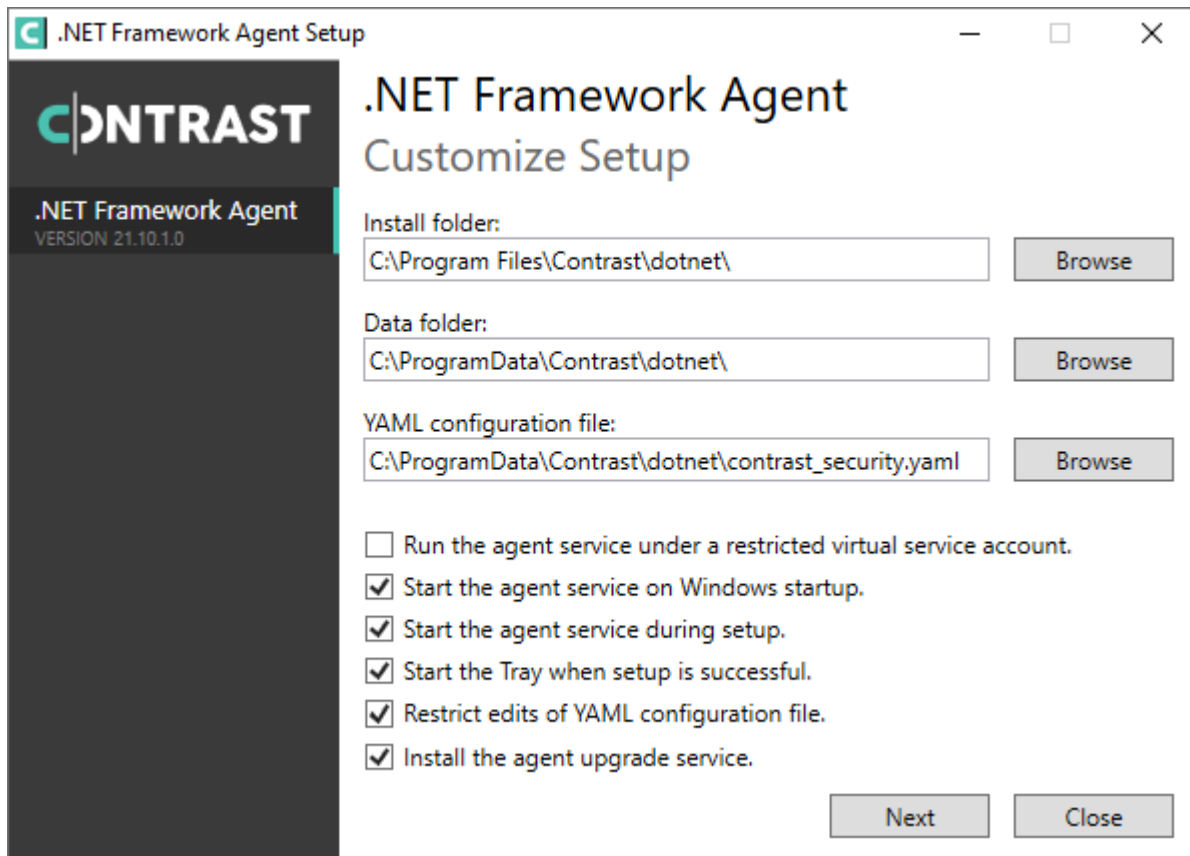


NOTE

When a new agent version installed, IIS will be restarted.

The Agent Upgrade Service is an optional component and is not required for agent Assess and Protect features.

- De-select the **Install the agent upgrade service** checkbox when installing the agent if you do not want to use the Agent Upgrade Service.
- If installing the agent via command line, add `INSTALL_UPGRADE_SERVICE=0` argument to not install the Agent Upgrade Service.



The behavior of the Agent Upgrade Service can be modified via an agent-specific configuration file in the Contrast data directory. The default location is `C:\ProgramData\Contrast\upgrade-service`.

The configuration for upgrading the .NET Core agent is located in the .NET Core YAML file.

```
enable: true # Set to `true` for the agent to automatically upgrade to
newer versions.
checks: Startup # Set the frequency with which the agent checks for
updates. Valid values are `daily` for every 24 hours and on startup, or
`startup` for *only* when service starts up.
timeout_ms: 60000 # Set the time allocated to execute the downloaded agent
installer before cancelling.
nuget_repository_url: https://api.nuget.org/v3/index.json # Set the URL of
the Nuget repository to be used for the .NET Core Agent for IIS Installer
nuget_package_name: Contrast.CoreIIS.Installer # Set the name of the .NET
Core Agent for IIS Nuget package.
installer_upgrade_code: 82468c04-dfc0-4a4c-9eb9-c4b314c67fdc # Used
internally to retrieve the current installed agent version from Windows.
enable_major_version_upgrade: false # Set to `true` to automatically
upgrade major versions.
```



NOTE

The Agent Upgrade Service is only included with the agent installer. It is not included with the manual .NET Core Agent, agent NuGet packages, or Azure App Service site extensions.

Update the .NET Framework agent

Use either of these methods to update the .NET Framework agent:

- Automatically update the agent
- Use the Contrast API to download the agent and install with the command line

Before you begin

- Confirm your .NET Framework application runs properly without the Contrast .NET Framework agent.
- Previously successfully installed the Contrast .NET Framework agent.
- Defined a policy for how and when to update the agent, based on your change management policy, workflow, and the environment where you deploy agents.
- Have some familiarity with Windows scripting methods, including PowerShell.

Update the agent automatically

The agent automatically updates through the [Agent upgrade service \(page 226\)](#). Contrast supports and releases new versions of the agent, versions 20.5.1 and later. The agent does not auto-update if it detects an [unsupported environment \(page 212\)](#).

Use the Contrast API to download the agent

1. Download the Contrast agent directly from the Contrast API. This step works for both hosted and on-premises instances of Contrast.
You need an account (service or user) to access the API.
2. Use the silent installer to install the agent with the command line, after downloading the agent.
You can find your credentials by [viewing your organization and personal keys \(page 598\)](#).

See also

- [Agent upgrade service \(page 226\)](#)

Configure the .NET Framework agent

The [standard configuration \(page 102\)](#) uses this [order of precedence \(page 106\)](#).

Configure the .NET Framework agent:

- [For Azure App Service \(page 228\)](#)
- [In the `web.config` file \(page 229\)](#)
- [With the .NET Framework YAML template \(page 230\)](#)



TIP

Use the [Contrast agent configuration editor \(page 109\)](#) to create or upload a YAML configuration file, validate YAML and get setting recommendations.

.NET Framework agent-specific settings for Azure App Service

You can configure the .NET Framework agent for Azure App Service in the Azure Portal in three ways:

- Use the environment variable convention of agent configuration. Add all settings to the **Application Settings** section of the **Configuration** blade in the Azure Portal using [environment variable syntax \(page 110\)](#).

- Specify application configuration options in an application's `web.config` file. For the agent to pick up customized application settings, you must place these settings in the application `web.config` file's root configuration `appSettings` section. See [application-specific settings for Windows \(page 229\)](#) for more details.
- Instead of setting individual options in the Azure Portal, you may use a YAML configuration file containing Contrast settings. First, upload the file to your Azure web application by including it in your application deployment or using the Kudu console. Then add an application setting, `CONTRAST_CONFIG_PATH`, that points to this file.

For example, To use the `contrast_security.yaml` file in the root of your application, add an application setting with key `CONTRAST_CONFIG_PATH` and value `D:\Home\site\wwwroot\contrast_security.yaml`. Application files in Azure App Service are deployed to `D:\home\site\wwwroot`.

Configure .NET Framework with web.config file

You can specify the configuration options in an application's `web.config` file or using YAML configuration. For the agent to pick up customized application settings with `web.config`, you must place these settings in the application `web.config` file's root configuration `appSettings` section.

For example, two applications hosted in the same application pool will report as different servers if you configure the `contrast.server.name` property in the `appSettings` in each application's `web.config` file. Or, you could use `web.config` to configure the `contrast.application.name`, like this:

```
<configuration>
  <appSettings>
    <add key="contrast.application.name" value="MyWebAppName" />
    <add key="contrast.application.version" value="1.2.3" />
  </appSettings>
  <system.web>
    ...
  </system.web>
</configuration>
```

See the [.NET Framework YAML template \(page 230\)](#) for a description of other available properties.

If your agent version is earlier than 21.1.4, only some properties can be configured with `web.config` as listed here.

Properties	Introduced with this .NET Framework agent version
<code>contrast.application.code</code>	19.6.3
<code>contrast.application.group</code>	19.1.3
<code>contrast.application.metadata</code>	19.1.3
<code>contrast.application.name</code>	19.1.3
<code>contrast.application.session_id</code>	20.6.6
<code>contrast.application.session_metadata</code>	20.6.6
<code>contrast.application.tags</code>	19.1.3
<code>contrast.application.version</code>	19.1.3
<code>contrast.assess.tags</code>	19.1.3
<code>contrast.inventory.tags</code>	19.1.3



NOTE

If `contrast.application.name` is not specified, the .NET Framework agent will use the application's virtual path as an application name. If the application is hosted in the root of a site (meaning, the virtual path is `/`), the .NET Framework agent will use the site's name as the application name.



IMPORTANT

Starting with agent version 21.1.4, users can set most agent configuration settings either with the application's `web.config` file or with a `contrast_security.yaml` file in the same directory as the application. For example, two applications hosted in the same application pool can now report as different servers by setting `contrast.server.name` in the `appSettings` in each application's `web.config` file.

The following configuration settings are applied at the *process* level and cannot be customized separately for each application. You cannot set these properties using *web.config* and must set these configurations another way (like with YAML).

- `agent.dotnet.app_pool_denylist`
- `agent.dotnet.app_pool_allowlist`
- `agent.dotnet.enable_instrumentation_optimizations`
- `agent.dotnet.enable_jit_inlining`
- `agent.dotnet.enable_transparency_checks`
- `agent.dotnet.enable_struct_dataflow`
- `assess.enable_control_detection`

Additionally, the agent's profiler component uses the process-level settings for the following keys, while the agent's sensor component will use the application-specific settings (if specified):

- `agent.logger.level`
- `agent.logger.stdout`

.NET Framework YAML template

Configure the .NET Framework agent using a [YAML configuration \(page 107\)](#) file.

The `contrast_security.yaml` file is copied to the agent's data directory by the installer (`C:\ProgramData\Contrast\dotnet\contrast_security.yaml` by default). The installer does not copy the YAML file if it already exists at the destination.

The template below contains all valid YAML options for this agent. For example, you can use the file to set the server name reported by the .NET Framework agent. To do this, update the `contrast_security.yaml` file, add a new line and the code below, and then continue the installation as normal.

```
server:
  name: MyServerName
```

```
#
```

```
=====
==
```

```
# Use the properties in this YAML file to configure a Contrast agent.
# Go to https://docs.contrastsecurity.com/en/order-of-precedence.html
# to determine the order of precedence for configuration values.
```

```
#
```

```
=====
==
```

```
# Use this setting if you want to temporarily disable a Contrast agent.
# Set to `true` to enable the agent; set to `false` to disable the agent.
# enable: true

#
=====
==
# api
# Use the properties in this section to connect the agent to the Contrast
# UI.
#
=====
==
api:

# ***** REQUIRED *****
# Set the URL for the Contrast UI.
url: https://app.contrastsecurity.com/Contrast

# ***** REQUIRED *****
# Set the API key needed to communicate with the Contrast UI.
api_key: NEEDS_TO_BE_SET

# ***** REQUIRED *****
# Set the service key needed to communicate with the Contrast
# UI. It is used to calculate the Authorization header.
service_key: NEEDS_TO_BE_SET

# ***** REQUIRED *****
# Set the user name used to communicate with the Contrast
# UI. It is used to calculate the Authorization header.
user_name: NEEDS_TO_BE_SET

# base64 encoded JSON object containing the `url`,
# `api_key`, `service_key`, and `user_name` config options,
# allowing them all to be set in a single variable.
# token: NEEDS_TO_BE_SET

# Set the version of the TLS protocol the agent uses to communicate with
the
# Contrast UI. The .NET agent default behavior is
(SecurityProtocolType.Tls
# | SecurityProtocolType.Tls11 | SecurityProtocolType.Tls12).
# tls_versions: tls|tls11|tls12

#
=====
# api.certificate
# Use the following properties for communication
# with the Contrast UI using certificates.
#
=====
# certificate:

# If set to `false`, the agent will ignore the
```

```
# certificate configuration in this section.
# enable: true

# Determine the location from which the agent loads a client
# certificate. Value options include `File` or `Store`.
# certificate_location: NEEDS_TO_BE_SET

# Set the absolute path to the client certificate's
# .CER file for communication with Contrast UI. The
# `certificate_location` property must be set to `File`.
# cer_file: NEEDS_TO_BE_SET

# Specify the name of certificate store to open. The
# `certificate_location` property must be set to `Store`.
# Value options include `AuthRoot`, `CertificateAuthority`,
# `My`, `Root`, `TrustedPeople`, or `TrustedPublisher`.
# store_name: NEEDS_TO_BE_SET

# Specify the location of the certificate store. The
# `certificate_location` property must be set to `Store`.
# Value options include `CurrentUser` or `LocalMachine`.
# store_location: NEEDS_TO_BE_SET

# Specify the type of value the agent uses to find the certificate
# in the collection of certificates from the certificate store.
# The `certificate_location` property must be set to `Store`.
# Value options include `FindByIssuerDistinguishedName`,
# `FindByIssuerName`, `FindBySerialNumber`,
# `FindBySubjectDistinguishedName`, `FindBySubjectKeyIdentifier`,
# `FindBySubjectName`, or `FindByThumbprint`.
# find_type: NEEDS_TO_BE_SET

# Specify the value the agent uses in combination with
# `find_type` to find a certification in the certificate store.
#
# Note - The agent will use the first certificate from
# the certificate store that matches this search criteria.
#
# find_value: NEEDS_TO_BE_SET
```

```
#
```

```
=====
# api.proxy
# Use the following properties for communication
# with the Contrast UI over a proxy.
#
```

```
=====
# proxy:
```

```
# Set value to `true` for the agent to communicate
# with the Contrast web interface over a proxy. Set
# value to `false` if you don't want to use the proxy.
# enable: NEEDS_TO_BE_SET

# Set the URL for your Proxy Server. The URL form is `scheme://`
```

```
host:port`.
# url: NEEDS_TO_BE_SET

# Set the proxy user.
# user: NEEDS_TO_BE_SET

# Set the proxy password.
# pass: NEEDS_TO_BE_SET

# Set the proxy authentication type. Value
# options are `NTLM`, `Digest`, and `Basic`.
# auth_type: NEEDS_TO_BE_SET

#
=====
==
# agent
# Use the properties in this section to control the way and frequency
# with which the agent communicates to logs and the Contrast UI.
#
=====
==
# agent:

#
=====
# agent.route_coverage
# Use the following properties for the route-based coverage feature.
#
=====
# route_coverage:

# Set to `false` to stop the agent from sending
# route-based coverage data to the Contrast UI when the
#
# application returns an error code indicating
# the request was not processed as expected.
#
# report_on_error: false

#
=====
# agent.logger
# Define the following properties to set logging values.
# If the following properties are not defined, the
# agent uses the logging values from the Contrast UI.
#
=====
# logger:

# Set the the log output level. Valid options are
# `ERROR`, `WARN`, `INFO`, `DEBUG`, and `TRACE`.
# level: INFO

# Set to `true` to redirect all logs to
```

```
# `stdout` instead of the file system.
# stdout: false

# Set the roll size for log files in megabytes. The agent will
# attempt to prevent the log file from being larger than this size.
# roll_size: 100

# Set the number of backup files to keep. Set to `0` to disable.
# backups: 10

#
=====
# agent.security_logger
# Define the following properties to set security logging
# values associated with Protect. If not defined, the agent
# uses the security logging (CEF) values from the Contrast UI.
#
=====
# security_logger:

# Set the log level for security logging. Valid options
# are `ERROR`, `WARN`, `INFO`, `DEBUG`, and `TRACE`.
# level: ERROR

#
=====
# agent.security_logger.syslog
# Define the following properties to set Syslog values. If the
properties
# are not defined, the agent uses the Syslog values from the Contrast
UI.
#
=====
# syslog:

# Set to `true` to enable Syslog logging.
# enable: NEEDS_TO_BE_SET

# Set the IP address of the Syslog server
# to which the agent should send messages.
# ip: NEEDS_TO_BE_SET

# Set the port of the Syslog server to
# which the agent should send messages.
# port: NEEDS_TO_BE_SET

# Set the facility code of the messages the agent sends to Syslog.
# facility: 19

# Set the log level of Exploited attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_exploited: ALERT

# Set the log level of Blocked attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
```

```
# severity_blocked: NOTICE

# Set the log level of Blocked At Perimeter
# attacks. Value options are `ALERT`, `CRITICAL`,
# `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_blocked_perimeter: NOTICE

# Set the log level of Probed attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_probed: WARNING

# Set the log level of Suspicious attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_suspicious: WARNING

# Set the connection type used for Syslog messages.
# Value options are `UNENCRYPTED` and `ENCRYPTED`.
# connection_type: UNENCRYPTED

#
=====
# agent.dotnet
# The following properties apply to any .NET agent-wide configurations.
#
=====
# dotnet:

# Set a list of application pool names that the agent does not
# instrument or analyze. Names must be formatted as a comma-separated
# list. New after .NET Framework 19.1.3 and .NET Core 4.0.2.
# app_pool_denylist: NEEDS_TO_BE_SET

# Set a list of application pool names that the agent instruments or
# analyzes. If set, other application pools are ignored. Allowlist takes
# precedence over denylist. Names must be formatted as a comma-separated
# list. New after .NET Framework 19.1.3 and .NET Core 4.0.2.
# app_pool_allowlist: NEEDS_TO_BE_SET

# Set a list of application names that the agent does not
# analyze. (The applications are still instrumented).
# Names must be formatted as a comma-separated list.
# New after .NET Framework 19.1.3 and .NET Core 1.0.0.
# application_denylist: NEEDS_TO_BE_SET

# Set a list of application names that the agent analyzes.
# If set, other applications are not analyzed, but are
# still instrumented. Allowlist takes precedence over
# denylist. Names must be formatted as a comma-separated
# list. New after .NET Framework 19.1.3 and .NET Core 1.0.0.
# application_allowlist: NEEDS_TO_BE_SET

# Enable a profiler chaining feature to allow Contrast to
# work alongside other tools that use the CLR Profiling
# API. Defaults to `true`. New after .NET Framework 19.1.3
# (Installed Only) and .NET Core 1.9.3 (Installed Only).
```

```
# enable_chaining: true

# Indicate that the agent should produce a report that
# summarizes application hosting on the server (e.g.,
# CLR versions, bitness or pipeline modes). Defaults to
# `true`. New after .NET Framework 19.1.3 (Installed Only).
# enable_dvnr: true

# Indicate that the agent should monitor configuration files for
# changes. New after .NET Framework 50.0.15 and .NET Core 2.1.14.
# enable_file_watching: true

# Indicate that the agent should allow CLR optimizations
# of JIT-compiled methods. Defaults to `true`. New
# after .NET Framework 19.1.3 and .NET Core 1.0.0.
# enable_instrumentation_optimizations: true

# Indicate that the agent should allow the CLR to inline
# methods that are not instrumented by Contrast. Defaults to
# `true`. New after .NET Framework 19.1.3 and .NET Core 1.0.0.
# enable_jit_inlining: true

# Indicate that the agent should allow the CLR to perform
# transparency checks under full trust. Defaults to `false`.
# New after .NET Framework 19.1.3 and .NET Core 1.0.0.
# enable_transparency_checks: false

# Indicate that the agent should report the full file path for
# vulnerabilities discovered via file analysis. Note that this may
# lead to duplicate reports if an application is deployed on multiple
# servers with different paths. New after .NET Framework 50.1.9.
# file_analysis_report_full_path: false

# Responses for request paths (e.g., HttpRequest.Path)
# that match this regex are not analyzed. Defaults to
# `WebResource.axd`. New after .NET Framework 19.1.3.
# web_module_allowlist: WebResource.axd

# Set to display ASCII art to std::out on agent startup. Defaults
# to `true`. New after .NET Framework 20.6.3 and .NET Core 1.0.0.
# enable_cat: true

# Sets the maximum amount of time a Protect regular expression
# is allowed to run before being cancelled. Set to -1 to never
# cancel regular expression execution. Defaults to `20_000`.
# New after .NET Framework 20.4.3 and .NET Core 1.5.0.
# protect_searchers_single_pattern_deadline_ms: 20_000

# Sets the maximum amount of time a 'Probe Analysis' Protect
# regular expression is allowed to run before being cancelled. Set
# to -1 to never cancel regular expression execution. Defaults to
# `5_000`. New after .NET Framework 20.7.3 and .NET Core 1.5.11.
# protect_searchers_probe_analysis_single_pattern_deadline_ms: 5_000

# Sets the maximum amount of time a Protect rule is
```



```
# allowed to run before being cancelled. Set to -1 to never
# cancel Protect rule execution. Defaults to `60_000`.
# New after .NET Framework 20.4.3 and .NET Core 1.5.0.
# protect_searchers_total_rule_deadline_ms: 60_000

# Sets the maximum amount of time a 'Probe Analysis' Protect
# rule is allowed to run before being cancelled. Set to -1 to
# never cancel Protect rule execution. Defaults to `10_000`.
# New after .NET Framework 20.7.3 and .NET Core 1.5.11.
# protect_searchers_probe_analysis_total_rule_deadline_ms: 10_000

# Sets the maximum duration of time agent log files should be kept
# since last write before being deleted by the agent. Defaults to
# `604_800_000`. New after .NET Framework 20.6.1 and .NET Core 1.5.5.
# log_cleanup_maximum_age_ms: 604_800_000

# Suppresses gathering process-level metrics (process level metrics are
# gathered by default), used to identify performance problems. Metric
# counters may further decrease the stability of already unstable
# systems and can be disabled (set to true) if issues occur. Defaults
# to `false`. New after .NET Framework 20.6.6 and .NET Core 1.5.10.
# suppress_metric_counters: false

# Enable file based application watching. Set to false if
# file watching is causing locking issues. Defaults to `true`.
# New after .NET Framework 20.7.3 and .NET Core 1.5.11.
# enable_file_based_app_watching: true

# Enables HttpClient isolation using AppDomain remoting. This can be
used
# to workaround .NET TLS version limitations at the cost of performance
# and stability. Enabled by default on applications targeting .NET
# Framework < 4.7.0, else disabled. New after .NET Framework 21.5.1.
# enable_http_client_app_domain_isolation: false

# Enables LINQ optimizations to improve performance
# at the cost of possible false negatives. Defaults
# to `true`. New after .NET Framework 50.0.1.
# enable_linq_optimizations: true

#
=====
# agent.dotnet.file_analysis_time_ms
# Controls the interval in milliseconds to perform file
# analysis for supported rules. Setting a value > 0 will
# result in the job running at that interval and not just when
# the application loads. If set to `-1`, the job just runs
# once. Defaults to `-1`. New after .NET Framework 50.0.15.
#
=====
# file_analysis_time_ms: {}

#
=====
==
```

```
# inventory
# Use the properties in this section to override the inventory features.
#
=====
==
# inventory:

# Set to `false` to disable inventory features in the agent.
# enable: true

# Apply a list of labels to libraries. Labels
# must be formatted as a comma-delimited list.
# Example - `label1, label2, label3`
#
# tags: NEEDS_TO_BE_SET

#
=====
==
# assess
# Use the properties in this section to control Assess.
#
=====
==
# assess:

# Include this property to determine if the Assess
# feature should be enabled. If this property is not
# present, the decision is delegated to the Contrast UI.
# enable: false

# Control the values captured by Assess vulnerability events. `Full`
# captures most values by calling ToString on objects, which can
# provide more info but causes increased memory usage. `Minimal`
# has better performance as it only captures String type objects
# as strings and uses type name for other object type values.
# event_detail: minimal

# Apply a list of labels to vulnerabilities and preflight
# messages. Labels must be formatted as a comma-delimited list.
# Example - `label1, label2, label3`
#
# tags: NEEDS_TO_BE_SET

# Value options are `ALL`, `SOME`, or `NONE`.
# stacktraces: ALL

#
=====
# assess.sampling
# Use the following properties to control sampling in the agent.
#
=====
# sampling:
```

```
# Set to `true` to enable sampling.
# enable: false

# This property indicates the number of requests
# to analyze in each window before sampling begins.
# baseline: 5

# This property indicates that every *nth*
# request after the baseline is analyzed.
# request_frequency: 10

# This property indicates the duration for which a sample set is valid.
# window_ms: 180_000

#
=====
# assess.rules
# Use the following properties to control simple rule configurations.
#
=====
# rules:

# Define a list of Assess rules to disable in the agent. To view a
# list of rule names, in Contrast go to user menu > Policy Management >
# Assess rules. The rules must be formatted as a comma-delimited list.
#
# Example - Set `reflected-xss,sql-injection` to disable
# the reflected-xss rule and the sql-injection rule.
#
# disabled_rules: NEEDS_TO_BE_SET

#
=====
==
# protect
# Use the properties in this section to override Protect features.
#
=====
==
# protect:

# Include this property to determine if the Protect
# feature should be enabled. If this property is not
# present, the decision is delegated to the Contrast UI.
# enable: false

#
=====
# protect.probe_analysis
# Use the settings in this section to
# control the behavior of probe analysis.
#
=====
# probe_analysis:
```

```
# Set to `false` to disable probe analysis.
# enable: true

#
=====
# protect.rules
# Use the following properties to set simple rule configurations.
#
=====
# rules:

# Define a list of Protect rules to disable in the agent. To view a
# list of rule names, in Contrast go to user menu > Policy Management >
# Protect rules. The rules must be formatted as a comma-delimited list.
# disabled_rules: NEEDS_TO_BE_SET

#
=====
# protect.rules.bot-blocker
# Use the following selection to configure if the
# agent blocks bots. Set to `true` to enable blocking.
#
=====
# bot-blocker:

# Set to `true` for the agent to block known bots.
# enable: false

#
=====
# protect.rules.sql-injection
# Use the following settings to configure the sql-injection rule.
#
=====
# sql-injection:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or off.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.sql-injection-semantic-chaining
# Use the following properties to configure how the
# sql injection semantic analysis chaining rule works.
#
=====
# sql-injection-semantic-chaining:

# Set the mode of the rule. Value options
# are `monitor`, `block` or `off`.
```

```
# mode: off

#
=====
# protect.rules.sql-injection-semantic-dangerous-functions
# Use the following properties to configure how the sql
# injection semantic analysis dangerous functions rule works.
#
=====
# sql-injection-semantic-dangerous-functions:

# Set the mode of the rule. Value options
# are `monitor`, `block` or `off`.
# mode: off

#
=====
# protect.rules.sql-injection-semantic-suspicious-unions
# Use the following properties to configure how the sql
# injection semantic analysis suspicious unions rule works.
#
=====
# sql-injection-semantic-suspicious-unions:

# Set the mode of the rule. Value options
# are `monitor`, `block` or `off`.
# mode: off

#
=====
# protect.rules.sql-injection-semantic-tautologies
# Use the following properties to configure how the sql
# injection semantic analysis tautologies rule works.
#
=====
# sql-injection-semantic-tautologies:

# Set the mode of the rule. Value options
# are `monitor`, `block` or `off`.
# mode: off

#
=====
# protect.rules.cmd-injection
# Use the following properties to configure
# how the command injection rule works.
#
=====
# cmd-injection:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
```

```
#
# mode: off

# Tell the agent to detect when commands come directly
# from input. The agent blocks if blocking is enabled.
# detect_phased_commands: true

#
=====
# protect.rules.cmd-injection-semantic-chained-commands
# Use the following properties to configure how the
# 'command injection - chained commands' rule works
#
=====
# cmd-injection-semantic-chained-commands:

# Set the mode of the rule. Value options
# are `monitor`, `block`, or `off`.
# mode: off

#
=====
# protect.rules.cmd-injection-semantic-dangerous-paths
# Use the following properties to configure how the
# 'command injection - dangerous paths' rule works
#
=====
# cmd-injection-semantic-dangerous-paths:

# Set the mode of the rule. Value options
# are `monitor`, `block`, or `off`.
# mode: off

#
=====
# protect.rules.cmd-injection-command-backdoors
# Use the following properties to configure how the
# 'command injection - command backdoors' rule works
#
=====
# cmd-injection-command-backdoors:

# Set the mode of the rule. Value options
# are `monitor`, `block`, or `off`.
# mode: off

#
=====
# protect.rules.path-traversal-semantic-file-security-bypass
# Use the following properties to configure how the
# 'path traversal - file security bypass' rule works
#
=====
# path-traversal-semantic-file-security-bypass:
```

```
# Set the mode of the rule. Value options
# are `monitor`, `block`, or `off`.
# mode: off

#
=====
# protect.rules.path-traversal
# Use the following properties to configure
# how the path traversal rule works.
#
=====
# path-traversal:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.method-tampering
# Use the following properties to configure
# how the method tampering rule works.
#
=====
# method-tampering:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.reflected-xss
# Use the following properties to configure how
# the reflected cross-site scripting rule works.
#
=====
# reflected-xss:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off
```

```
#
=====
# protect.rules.unsafe-file-upload
# Use the following properties to configure
# how the unsafe file upload rule works.
#
=====
# unsafe-file-upload:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.xxe
# Use the following properties to configure
# how the XML external entity works.
#
=====
# xxe:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.untrusted-deserialization
# Use the following properties to configure
# how the untrusted deserialization rule works.
#
=====
# untrusted-deserialization:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
==
# application
```



```
# Use the properties in this section for
# the application(s) hosting this agent.
#
=====
==
# application:

# Override the reported application name.
#
# Note - On systems where multiple, distinct applications may be served
# by a single process, this configuration causes the agent to report
# all discovered applications as one application with the given name.
#
# Note - On Java systems where multiple, distinct applications may be
# served by a single process, this configuration causes the agent to
report
# all discovered applications as one application with the given name.
#
# name: NEEDS_TO_BE_SET

# Add the name of the application group with which this
# application should be associated in the Contrast UI.
# group: NEEDS_TO_BE_SET

# Add the application code this application should use in the Contrast UI.
# code: NEEDS_TO_BE_SET

# Override the reported application version.
# version: NEEDS_TO_BE_SET

# Apply labels to an application. Labels must
# be formatted as a comma-delimited list.
# Example - `label1,label2,label3`
#
# tags: NEEDS_TO_BE_SET

# Define a set of `key=value` pairs (which conforms to RFC 2253) for
# specifying user-defined metadata associated with the application. The
# set must be formatted as a comma-delimited list of `key=value` pairs.
# Example - `business-unit=accounting, office=Baltimore`
#
# metadata: NEEDS_TO_BE_SET

# Provide the ID of a session which already exists in the Contrast
# UI. Vulnerabilities discovered by the agent are associated with
# this session. If an invalid ID is supplied, the agent will be
# disabled. This option and `application.session_metadata` are
# mutually exclusive; if both are set, the agent will be disabled.
# session_id: NEEDS_TO_BE_SET

# Provide metadata which is used to create a new session ID in the
# Contrast UI. Vulnerabilities discovered by the agent are associated with
# this new session. This value should be formatted as `key=value` pairs
# (conforming to RFC 2253). Available key names for this configuration
# are branchName, buildNumber, commitHash, committer, gitTag, repository,
```

```
# testRun, and version. This option and `application.session_id` are
# mutually exclusive; if both are set the agent will be disabled.
# session_metadata: NEEDS_TO_BE_SET

#
=====
==
# server
# Use the settings in this section to set metadata for the server
# hosting this agent. Contrast recognizes common, supported server
# names, paths, types and environments. Doing this may require a new
# server or license, and it may affect functionality of some features.
#
=====
==
# server:

# Override the reported server name.
# name: localhost

# Set the environment directly to override the default set
# by the Contrast UI. This allows the user to configure the
# environment dynamically at startup rather than manually
# updating the Server in the Contrast UI themselves afterwards.
#
# Valid values include `QA`, `PRODUCTION` and `DEVELOPMENT`.
# For example, `PRODUCTION` registers this Server as
# running in a `PRODUCTION` environment, regardless of the
# organization's default environment in the Contrast UI.
#
# environment: NEEDS_TO_BE_SET

# Apply a list of labels to the server. Labels
# must be formatted as a comma-delimited list.
# Example - `label1,label2,label3`
#
# tags: NEEDS_TO_BE_SET

# Override the reported server path. New after
# .NET Framework v21.3.1 and .NET Core v1.8.0.
# path: NEEDS_TO_BE_SET

#
=====
==
# Use the properties in this YAML file to configure a Contrast agent.
# Go to https://docs.contrastsecurity.com/en/order-of-precedence.html
# to determine the order of precedence for configuration values.
#
=====
==

# Use this setting if you want to temporarily disable a Contrast agent.
# Set to `true` to enable the agent; set to `false` to disable the agent.
```

```
# enable: true

#
=====
==
# api
# Use the properties in this section to connect the agent to the Contrast
# UI.
#
=====
==
api:

# ***** REQUIRED *****
# Set the URL for the Contrast UI.
url: https://app.contrastsecurity.com/Contrast

# ***** REQUIRED *****
# Set the API key needed to communicate with the Contrast UI.
api_key: NEEDS_TO_BE_SET

# ***** REQUIRED *****
# Set the service key needed to communicate with the Contrast
# UI. It is used to calculate the Authorization header.
service_key: NEEDS_TO_BE_SET

# ***** REQUIRED *****
# Set the user name used to communicate with the Contrast
# UI. It is used to calculate the Authorization header.
user_name: NEEDS_TO_BE_SET

# base64 encoded JSON object containing the `url`,
# `api_key`, `service_key`, and `user_name` config options,
# allowing them all to be set in a single variable.
# token: NEEDS_TO_BE_SET

# Set the version of the TLS protocol the agent uses to communicate with
the
# Contrast UI. The .NET agent default behavior is
(SecurityProtocolType.Tls
# | SecurityProtocolType.Tls11 | SecurityProtocolType.Tls12).
# tls_versions: tls|tls11|tls12

#
=====
# api.certificate
# Use the following properties for communication
# with the Contrast UI using certificates.
#
=====
# certificate:

# If set to `false`, the agent will ignore the
# certificate configuration in this section.
# enable: true
```

```
# Determine the location from which the agent loads a client
# certificate. Value options include `File` or `Store`.
# certificate_location: NEEDS_TO_BE_SET

# Set the absolute path to the client certificate's
# .CER file for communication with Contrast UI. The
# `certificate_location` property must be set to `File`.
# cer_file: NEEDS_TO_BE_SET

# Specify the name of certificate store to open. The
# `certificate_location` property must be set to `Store`.
# Value options include `AuthRoot`, `CertificateAuthority`,
# `My`, `Root`, `TrustedPeople`, or `TrustedPublisher`.
# store_name: NEEDS_TO_BE_SET

# Specify the location of the certificate store. The
# `certificate_location` property must be set to `Store`.
# Value options include `CurrentUser` or `LocalMachine`.
# store_location: NEEDS_TO_BE_SET

# Specify the type of value the agent uses to find the certificate
# in the collection of certificates from the certificate store.
# The `certificate_location` property must be set to `Store`.
# Value options include `FindByIssuerDistinguishedName`,
# `FindByIssuerName`, `FindBySerialNumber`,
# `FindBySubjectDistinguishedName`, `FindBySubjectKeyIdentifier`,
# `FindBySubjectName`, or `FindByThumbprint`.
# find_type: NEEDS_TO_BE_SET

# Specify the value the agent uses in combination with
# `find_type` to find a certification in the certificate store.
#
# Note - The agent will use the first certificate from
# the certificate store that matches this search criteria.
#
# find_value: NEEDS_TO_BE_SET

#
=====
# api.proxy
# Use the following properties for communication
# with the Contrast UI over a proxy.
#
=====
# proxy:

# Set value to `true` for the agent to communicate
# with the Contrast web interface over a proxy. Set
# value to `false` if you don't want to use the proxy.
# enable: NEEDS_TO_BE_SET

# Set the URL for your Proxy Server. The URL form is `scheme://
host:port`.
# url: NEEDS_TO_BE_SET
```

```
# Set the proxy user.
# user: NEEDS_TO_BE_SET

# Set the proxy password.
# pass: NEEDS_TO_BE_SET

# Set the proxy authentication type. Value
# options are `NTLM`, `Digest`, and `Basic`.
# auth_type: NEEDS_TO_BE_SET

#
=====
==
# agent
# Use the properties in this section to control the way and frequency
# with which the agent communicates to logs and the Contrast UI.
#
=====
==
# agent:

#
=====
# agent.route_coverage
# Use the following properties for the route-based coverage feature.
#
=====
# route_coverage:

# Set to `false` to stop the agent from sending
# route-based coverage data to the Contrast UI when the
#
# application returns an error code indicating
# the request was not processed as expected.
#
# report_on_error: false

#
=====
# agent.logger
# Define the following properties to set logging values.
# If the following properties are not defined, the
# agent uses the logging values from the Contrast UI.
#
=====
# logger:

# Set the the log output level. Valid options are
# `ERROR`, `WARN`, `INFO`, `DEBUG`, and `TRACE`.
# level: INFO

# Set to `true` to redirect all logs to
# `stdout` instead of the file system.
# stdout: false
```

```
# Set the roll size for log files in megabytes. The agent will
# attempt to prevent the log file from being larger than this size.
# roll_size: 100

# Set the number of backup files to keep. Set to `0` to disable.
# backups: 10

#
=====
# agent.security_logger
# Define the following properties to set security logging
# values associated with Protect. If not defined, the agent
# uses the security logging (CEF) values from the Contrast UI.
#
=====
# security_logger:

# Set the log level for security logging. Valid options
# are `ERROR`, `WARN`, `INFO`, `DEBUG`, and `TRACE`.
# level: ERROR

#
=====
# agent.security_logger.syslog
# Define the following properties to set Syslog values. If the
properties
# are not defined, the agent uses the Syslog values from the Contrast
UI.
#
=====
# syslog:

# Set to `true` to enable Syslog logging.
# enable: NEEDS_TO_BE_SET

# Set the IP address of the Syslog server
# to which the agent should send messages.
# ip: NEEDS_TO_BE_SET

# Set the port of the Syslog server to
# which the agent should send messages.
# port: NEEDS_TO_BE_SET

# Set the facility code of the messages the agent sends to Syslog.
# facility: 19

# Set the log level of Exploited attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_exploited: ALERT

# Set the log level of Blocked attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_blocked: NOTICE
```

```
# Set the log level of Blocked At Perimeter
# attacks. Value options are `ALERT`, `CRITICAL`,
# `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_blocked_perimeter: NOTICE

# Set the log level of Probed attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_probed: WARNING

# Set the log level of Suspicious attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_suspicious: WARNING

# Set the connection type used for Syslog messages.
# Value options are `UNENCRYPTED` and `ENCRYPTED`.
# connection_type: UNENCRYPTED

#
=====
# agent.dotnet
# The following properties apply to any .NET agent-wide configurations.
#
=====
# dotnet:

# Set a list of application pool names that the agent does not
# instrument or analyze. Names must be formatted as a comma-separated
# list. New after .NET Framework 19.1.3 and .NET Core 4.0.2.
# app_pool_denylist: NEEDS_TO_BE_SET

# Set a list of application pool names that the agent instruments or
# analyzes. If set, other application pools are ignored. Allowlist takes
# precedence over denylist. Names must be formatted as a comma-separated
# list. New after .NET Framework 19.1.3 and .NET Core 4.0.2.
# app_pool_allowlist: NEEDS_TO_BE_SET

# Set a list of application names that the agent does not
# analyze. (The applications are still instrumented).
# Names must be formatted as a comma-separated list.
# New after .NET Framework 19.1.3 and .NET Core 1.0.0.
# application_denylist: NEEDS_TO_BE_SET

# Set a list of application names that the agent analyzes.
# If set, other applications are not analyzed, but are
# still instrumented. Allowlist takes precedence over
# denylist. Names must be formatted as a comma-separated
# list. New after .NET Framework 19.1.3 and .NET Core 1.0.0.
# application_allowlist: NEEDS_TO_BE_SET

# Enable a profiler chaining feature to allow Contrast to
# work alongside other tools that use the CLR Profiling
# API. Defaults to `true`. New after .NET Framework 19.1.3
# (Installed Only) and .NET Core 1.9.3 (Installed Only).
# enable_chaining: true
```

```
# Indicate that the agent should produce a report that
# summarizes application hosting on the server (e.g.,
# CLR versions, bitness or pipeline modes). Defaults to
# `true`. New after .NET Framework 19.1.3 (Installed Only).
# enable_dvnr: true

# Indicate that the agent should monitor configuration files for
# changes. New after .NET Framework 50.0.15 and .NET Core 2.1.14.
# enable_file_watching: true

# Indicate that the agent should allow CLR optimizations
# of JIT-compiled methods. Defaults to `true`. New
# after .NET Framework 19.1.3 and .NET Core 1.0.0.
# enable_instrumentation_optimizations: true

# Indicate that the agent should allow the CLR to inline
# methods that are not instrumented by Contrast. Defaults to
# `true`. New after .NET Framework 19.1.3 and .NET Core 1.0.0.
# enable_jit_inlining: true

# Indicate that the agent should allow the CLR to perform
# transparency checks under full trust. Defaults to `false`.
# New after .NET Framework 19.1.3 and .NET Core 1.0.0.
# enable_transparency_checks: false

# Indicate that the agent should report the full file path for
# vulnerabilities discovered via file analysis. Note that this may
# lead to duplicate reports if an application is deployed on multiple
# servers with different paths. New after .NET Framework 50.1.9.
# file_analysis_report_full_path: false

# Responses for request paths (e.g., HttpRequest.Path)
# that match this regex are not analyzed. Defaults to
# `WebResource.axd`. New after .NET Framework 19.1.3.
# web_module_allowlist: WebResource.axd

# Set to display ASCII art to std::out on agent startup. Defaults
# to `true`. New after .NET Framework 20.6.3 and .NET Core 1.0.0.
# enable_cat: true

# Sets the maximum amount of time a Protect regular expression
# is allowed to run before being cancelled. Set to -1 to never
# cancel regular expression execution. Defaults to `20_000`.
# New after .NET Framework 20.4.3 and .NET Core 1.5.0.
# protect_searchers_single_pattern_deadline_ms: 20_000

# Sets the maximum amount of time a 'Probe Analysis' Protect
# regular expression is allowed to run before being cancelled. Set
# to -1 to never cancel regular expression execution. Defaults to
# `5_000`. New after .NET Framework 20.7.3 and .NET Core 1.5.11.
# protect_searchers_probe_analysis_single_pattern_deadline_ms: 5_000

# Sets the maximum amount of time a Protect rule is
# allowed to run before being cancelled. Set to -1 to never
# cancel Protect rule execution. Defaults to `60_000`.
```



```
# New after .NET Framework 20.4.3 and .NET Core 1.5.0.
# protect_searchers_total_rule_deadline_ms: 60_000

# Sets the maximum amount of time a 'Probe Analysis' Protect
# rule is allowed to run before being cancelled. Set to -1 to
# never cancel Protect rule execution. Defaults to `10_000`.
# New after .NET Framework 20.7.3 and .NET Core 1.5.11.
# protect_searchers_probe_analysis_total_rule_deadline_ms: 10_000

# Sets the maximum duration of time agent log files should be kept
# since last write before being deleted by the agent. Defaults to
# `604_800_000`. New after .NET Framework 20.6.1 and .NET Core 1.5.5.
# log_cleanup_maximum_age_ms: 604_800_000

# Suppresses gathering process-level metrics (process level metrics are
# gathered by default), used to identify performance problems. Metric
# counters may further decrease the stability of already unstable
# systems and can be disabled (set to true) if issues occur. Defaults
# to `false`. New after .NET Framework 20.6.6 and .NET Core 1.5.10.
# suppress_metric_counters: false

# Enable file based application watching. Set to false if
# file watching is causing locking issues. Defaults to `true`.
# New after .NET Framework 20.7.3 and .NET Core 1.5.11.
# enable_file_based_app_watching: true

# Enables HttpClient isolation using AppDomain remoting. This can be
used
# to workaround .NET TLS version limitations at the cost of performance
# and stability. Enabled by default on applications targeting .NET
# Framework < 4.7.0, else disabled. New after .NET Framework 21.5.1.
# enable_http_client_app_domain_isolation: false

# Enables LINQ optimizations to improve performance
# at the cost of possible false negatives. Defaults
# to `true`. New after .NET Framework 50.0.1.
# enable_linq_optimizations: true

#
=====
# agent.dotnet.file_analysis_time_ms
# Controls the interval in milliseconds to perform file
# analysis for supported rules. Setting a value > 0 will
# result in the job running at that interval and not just when
# the application loads. If set to `-1`, the job just runs
# once. Defaults to `-1`. New after .NET Framework 50.0.15.
#
=====
# file_analysis_time_ms: {}

#
=====
==
# inventory
# Use the properties in this section to override the inventory features.
```

```
#
=====
==
# inventory:

# Set to `false` to disable inventory features in the agent.
# enable: true

# Apply a list of labels to libraries. Labels
# must be formatted as a comma-delimited list.
# Example - `label1, label2, label3`
#
# tags: NEEDS_TO_BE_SET

#
=====
==
# assess
# Use the properties in this section to control Assess.
#
=====
==
# assess:

# Include this property to determine if the Assess
# feature should be enabled. If this property is not
# present, the decision is delegated to the Contrast UI.
# enable: false

# Control the values captured by Assess vulnerability events. `Full`
# captures most values by calling ToString on objects, which can
# provide more info but causes increased memory usage. `Minimal`
# has better performance as it only captures String type objects
# as strings and uses type name for other object type values.
# event_detail: minimal

# Apply a list of labels to vulnerabilities and preflight
# messages. Labels must be formatted as a comma-delimited list.
# Example - `label1, label2, label3`
#
# tags: NEEDS_TO_BE_SET

# Value options are `ALL`, `SOME`, or `NONE`.
# stacktraces: ALL

#
=====
# assess.sampling
# Use the following properties to control sampling in the agent.
#
=====
# sampling:

# Set to `true` to enable sampling.
# enable: false
```

```
# This property indicates the number of requests
# to analyze in each window before sampling begins.
# baseline: 5

# This property indicates that every *nth*
# request after the baseline is analyzed.
# request_frequency: 10

# This property indicates the duration for which a sample set is valid.
# window_ms: 180_000

#
=====
# assess.rules
# Use the following properties to control simple rule configurations.
#
=====
# rules:

# Define a list of Assess rules to disable in the agent. To view a
# list of rule names, in Contrast go to user menu > Policy Management >
# Assess rules. The rules must be formatted as a comma-delimited list.
#
# Example - Set `reflected-xss,sql-injection` to disable
# the reflected-xss rule and the sql-injection rule.
#
# disabled_rules: NEEDS_TO_BE_SET

#
=====
==
# protect
# Use the properties in this section to override Protect features.
#
=====
==
# protect:

# Include this property to determine if the Protect
# feature should be enabled. If this property is not
# present, the decision is delegated to the Contrast UI.
# enable: false

#
=====
# protect.probe_analysis
# Use the settings in this section to
# control the behavior of probe analysis.
#
=====
# probe_analysis:

# Set to `false` to disable probe analysis.
# enable: true
```

```
#
=====
# protect.rules
# Use the following properties to set simple rule configurations.
#
=====
# rules:

# Define a list of Protect rules to disable in the agent. To view a
# list of rule names, in Contrast go to user menu > Policy Management >
# Protect rules. The rules must be formatted as a comma-delimited list.
# disabled_rules: NEEDS_TO_BE_SET

#
=====
# protect.rules.bot-blocker
# Use the following selection to configure if the
# agent blocks bots. Set to `true` to enable blocking.
#
=====
# bot-blocker:

# Set to `true` for the agent to block known bots.
# enable: false

#
=====
# protect.rules.sql-injection
# Use the following settings to configure the sql-injection rule.
#
=====
# sql-injection:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or off.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.sql-injection-semantic-chaining
# Use the following properties to configure how the
# sql injection semantic analysis chaining rule works.
#
=====
# sql-injection-semantic-chaining:

# Set the mode of the rule. Value options
# are `monitor`, `block` or `off`.
# mode: off
```

```
#
=====
# protect.rules.sql-injection-semantic-dangerous-functions
# Use the following properties to configure how the sql
# injection semantic analysis dangerous functions rule works.
#
=====
# sql-injection-semantic-dangerous-functions:

# Set the mode of the rule. Value options
# are `monitor`, `block` or `off`.
# mode: off

#
=====
# protect.rules.sql-injection-semantic-suspicious-unions
# Use the following properties to configure how the sql
# injection semantic analysis suspicious unions rule works.
#
=====
# sql-injection-semantic-suspicious-unions:

# Set the mode of the rule. Value options
# are `monitor`, `block` or `off`.
# mode: off

#
=====
# protect.rules.sql-injection-semantic-tautologies
# Use the following properties to configure how the sql
# injection semantic analysis tautologies rule works.
#
=====
# sql-injection-semantic-tautologies:

# Set the mode of the rule. Value options
# are `monitor`, `block` or `off`.
# mode: off

#
=====
# protect.rules.cmd-injection
# Use the following properties to configure
# how the command injection rule works.
#
=====
# cmd-injection:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off
```

```
# Tell the agent to detect when commands come directly
# from input. The agent blocks if blocking is enabled.
# detect_phased_commands: true

#
=====
# protect.rules.cmd-injection-semantic-chained-commands
# Use the following properties to configure how the
# 'command injection - chained commands' rule works
#
=====
# cmd-injection-semantic-chained-commands:

# Set the mode of the rule. Value options
# are `monitor`, `block`, or `off`.
# mode: off

#
=====
# protect.rules.cmd-injection-semantic-dangerous-paths
# Use the following properties to configure how the
# 'command injection - dangerous paths' rule works
#
=====
# cmd-injection-semantic-dangerous-paths:

# Set the mode of the rule. Value options
# are `monitor`, `block`, or `off`.
# mode: off

#
=====
# protect.rules.cmd-injection-command-backdoors
# Use the following properties to configure how the
# 'command injection - command backdoors' rule works
#
=====
# cmd-injection-command-backdoors:

# Set the mode of the rule. Value options
# are `monitor`, `block`, or `off`.
# mode: off

#
=====
# protect.rules.path-traversal-semantic-file-security-bypass
# Use the following properties to configure how the
# 'path traversal - file security bypass' rule works
#
=====
# path-traversal-semantic-file-security-bypass:

# Set the mode of the rule. Value options
# are `monitor`, `block`, or `off`.
```

```
# mode: off

#
=====
# protect.rules.path-traversal
# Use the following properties to configure
# how the path traversal rule works.
#
=====
# path-traversal:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.method-tampering
# Use the following properties to configure
# how the method tampering rule works.
#
=====
# method-tampering:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.reflected-xss
# Use the following properties to configure how
# the reflected cross-site scripting rule works.
#
=====
# reflected-xss:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
```

```
# protect.rules.unsafe-file-upload
# Use the following properties to configure
# how the unsafe file upload rule works.
#
=====
# unsafe-file-upload:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.xxe
# Use the following properties to configure
# how the XML external entity works.
#
=====
# xxe:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.untrusted-deserialization
# Use the following properties to configure
# how the untrusted deserialization rule works.
#
=====
# untrusted-deserialization:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
==
# application
# Use the properties in this section for
# the application(s) hosting this agent.
```



```
#
=====
==
# application:

# Override the reported application name.
#
# Note - On systems where multiple, distinct applications may be served
# by a single process, this configuration causes the agent to report
# all discovered applications as one application with the given name.
#
# Note - On Java systems where multiple, distinct applications may be
# served by a single process, this configuration causes the agent to
report
# all discovered applications as one application with the given name.
#
# name: NEEDS_TO_BE_SET

# Add the name of the application group with which this
# application should be associated in the Contrast UI.
# group: NEEDS_TO_BE_SET

# Add the application code this application should use in the Contrast UI.
# code: NEEDS_TO_BE_SET

# Override the reported application version.
# version: NEEDS_TO_BE_SET

# Apply labels to an application. Labels must
# be formatted as a comma-delimited list.
# Example - `label1,label2,label3`
#
# tags: NEEDS_TO_BE_SET

# Define a set of `key=value` pairs (which conforms to RFC 2253) for
# specifying user-defined metadata associated with the application. The
# set must be formatted as a comma-delimited list of `key=value` pairs.
# Example - `business-unit=accounting, office=Baltimore`
#
# metadata: NEEDS_TO_BE_SET

# Provide the ID of a session which already exists in the Contrast
# UI. Vulnerabilities discovered by the agent are associated with
# this session. If an invalid ID is supplied, the agent will be
# disabled. This option and `application.session_metadata` are
# mutually exclusive; if both are set, the agent will be disabled.
# session_id: NEEDS_TO_BE_SET

# Provide metadata which is used to create a new session ID in the
# Contrast UI. Vulnerabilities discovered by the agent are associated with
# this new session. This value should be formatted as `key=value` pairs
# (conforming to RFC 2253). Available key names for this configuration
# are branchName, buildNumber, commitHash, committer, gitTag, repository,
# testRun, and version. This option and `application.session_id` are
# mutually exclusive; if both are set the agent will be disabled.
```

```
# session_metadata: NEEDS_TO_BE_SET

#
=====
==
# server
# Use the settings in this section to set metadata for the server
# hosting this agent. Contrast recognizes common, supported server
# names, paths, types and environments. Doing this may require a new
# server or license, and it may affect functionality of some features.
#
=====
==
# server:

# Override the reported server name.
# name: localhost

# Set the environment directly to override the default set
# by the Contrast UI. This allows the user to configure the
# environment dynamically at startup rather than manually
# updating the Server in the Contrast UI themselves afterwards.
#
# Valid values include `QA`, `PRODUCTION` and `DEVELOPMENT`.
# For example, `PRODUCTION` registers this Server as
# running in a `PRODUCTION` environment, regardless of the
# organization's default environment in the Contrast UI.
#
# environment: NEEDS_TO_BE_SET

# Apply a list of labels to the server. Labels
# must be formatted as a comma-delimited list.
# Example - `label1,label2,label3`
#
# tags: NEEDS_TO_BE_SET

# Override the reported server path. New after
# .NET Framework v21.3.1 and .NET Core v1.8.0.
# path: NEEDS_TO_BE_SET
```

Certificate exceptions

If you see certificate exception messages and feel that it's safe to ignore them, add this setting to the YAML configuration file:

```
api:
  certificate:
    ignore_cert_errors: true
```

🔗 [Learn more](#) about managing certificate issues.

Use the .NET Framework agent with applications on Azure

Use the Contrast .NET Framework agent to analyze ASP.NET applications running on Azure Virtual Machines (VMs), Azure Cloud Services, Mobile Services or Azure App Service (formerly Azure Web Sites).

To install the .NET Framework agent on Azure Virtual Machines:

1. Set up the Azure VM or the Azure Cloud Services as you would normally, and deploy the ASP.NET applications to be analyzed.
2. Log in to Contrast, and download the ZIP file for the .NET Framework agent.
3. Access the Remote Desktop [Azure VM](#) or [Azure Cloud Service](#) instance.
4. Copy the .NET Framework agent ZIP file to the Azure VM or to the Azure Cloud Services instance, and extract the archive.
5. Run the .NET Framework agent installer (*ContrastSetup.exe*).
6. Exercise the application so that Contrast can analyze it.



TIP

To install with Azure App Service (formerly Azure Web Apps), install with [NuGet \(page 223\)](#) or [Azure Portal Extension \(page 217\)](#).

Use Azure Service Fabric with the .NET Framework or .NET Core agent

If you are using a container image, follow the instructions to [install in containers \(page 220\)](#). Otherwise, to add the Contrast .NET Framework or .NET Core agent to an Azure Service Fabric service:



TIP

For Standalone Executable services, the `ServiceManifest.xml` file is located in the top-level Azure Service Fabric project (for example, the `sfproj` file).

1. Install the appropriate NuGet package to the main project for the service.
 - **.NET Framework:** Install `Contrast.NET.Azure.AppService`. All files in the `contrastsecurity` folder must have Copy to Output Directory set to Copy if newer.
 - **.NET Core:** Install `Contrast.SensorsNetCore`. All files in the `contrast` folder have Copy to Output Directory set to Copy if newer.
2. Set `ServiceManifest/CodePackage/EntryPoint/ExeHost/WorkingDirectory` in `ServiceManifest.xml` to `CodePackage`.

```
<CodePackage Name="Code" Version="1.0.0">
  <EntryPoint>
    <ExeHost>
      <Program>DemoNetFxStatelessService.exe</Program>
      <WorkingFolder>CodePackage</WorkingFolder>
    </ExeHost>
  </EntryPoint>
</CodePackage>
```

3. Set environment variables in `ServiceManifest.xml` to configure the profiler.
 - **.NET Framework:**

```
<CodePackage>
  <EnvironmentVariables>
    <EnvironmentVariable Name="COR_ENABLE_PROFILING"
Value="1"/>
    <EnvironmentVariable Name="COR_PROFILER"
```

```
Value="{EFEB8EE0-6D39-4347-A5FE-4D0C88BC5BC1}"/>
  <EnvironmentVariable
Name="COR_PROFILER_PATH_32" Value=".\contrastsecurity\runtimes\win-
x86\native\ContrastProfiler.dll" />
  <EnvironmentVariable
Name="COR_PROFILER_PATH_64" Value=".\contrastsecurity\runtimes\win-
x64\native\ContrastProfiler.dll" />
  <EnvironmentVariable Name="CONTRAST_CONFIG_PATH"
Value="contrast_security.yaml"/>
```

- **.NET Core:**

```
<CodePackage>
  <EnvironmentVariables>
    <EnvironmentVariable Name="CORECLR_ENABLE_PROFILING"
Value="1"/>
    <EnvironmentVariable Name="CORECLR_PROFILER"
Value="{8B2CE134-0948-48CA-A4B2-80DDAD9F5791}"/>
    <EnvironmentVariable Name="CORECLR_PROFILER_PATH_32"
Value="contrast\runtimes\win-x86\native\ContrastProfiler.dll"/>
    <EnvironmentVariable Name="CORECLR_PROFILER_PATH_64"
Value="contrast\runtimes\win-x64\native\ContrastProfiler.dll"/>
    <EnvironmentVariable Name="CONTRAST_CONFIG_PATH"
Value="contrast_security.yaml"/>
```

4. Configure the agent with either:

- **A YAML file:** Add it to the main project for the service. Make sure Copy to Output Directory for the file is set to Copy if newer. Add an environment variable to ServiceManifest.xml specifying the location of the file, like this:

```
<CodePackage>
  <EnvironmentVariables>
    <EnvironmentVariable Name="CONTRAST_CONFIG_PATH"
Value="contrast_security.yaml"/>
```

- **Environment variables:** Add them to ServiceManifest.xml, like this:

```
<CodePackage>
  <EnvironmentVariables>
    <EnvironmentVariable Name="CONTRAST__API__URL"
Value="https://teamserver-staging.contsec.com"/>
    <EnvironmentVariable Name="CONTRAST__API__API_KEY"
Value="aBcD0123"/>
    <EnvironmentVariable Name="CONTRAST__API__SERVICE_KEY"
Value="ABCD0123"/>
    <EnvironmentVariable Name="CONTRAST__API__USER_NAME"
Value="agent_123@Team"/>
```

5. Deploy the Azure Service Fabric application as usual.

Profiler chaining for the .NET Framework agent

You can use profiler chaining to run the .NET Framework agent alongside other .NET profiler agents, such as performance or APM tools.

The Contrast .NET Framework agent is tested and proven to be compatible with these profiling tools:

Profiling tool	Versions tested
AppDynamics	4.5.18.1
Dynatrace One Agent	1.253.245

Profiling tool	Versions tested
New Relic	8.23.107
Riverbed SteelCentral Aternity APM	12.9.0
Datadog	2.35.0

**NOTE**

The agent may also be compatible with other profiling tools if those tools follow the conventions of the CLR Profiling API and do not make assumptions about the profiling environment.

Profiler chaining is enabled by default. To disable profiler chaining, [configure the .NET Framework agent \(page 228\)](#) so that the `agent.dotnet.enable_chaining` setting is set to `false`. For example, you could use this YAML configuration:

```
agent:
  dotnet:
    enable_chaining: false
```

Once configured, you must restart the agent. On restart, the Contrast .NET Framework agent automatically detects the presence of other profiling tools registered with IIS and configures the environment to load both the Contrast .NET Framework agent profiler and the third-party profiler.



IMPORTANT

If you are using profiler chaining with:

- **IIS:**

Install the third-party agent, then the Contrast .NET Framework agent.
(If you install the Contrast .NET Framework agent before the third-party agent, you need to restart the Contrast.NET Main Service under **Windows Services**.)

- **Outside of IIS:**

The `agent.dotnet.enable_chaining` configuration flag will not work if you are using profiler chaining for applications:

- hosted outside of IIS, or
- that use the third-party agent's Nuget package (rather than the installed agent).

If this applies to you, replace the CLR environment variables for the profiling tool with `CONTRAST_CCC_COR` versions. Any of these names should be transformed:

Change this	To this
<code>COR_PROFILER</code>	<code>CONTRAST_CCC_COR_PROFILER</code>
<code>COR_PROFILER_PATH</code>	<code>CONTRAST_CCC_COR_PROFILER_PATH</code>
<code>COR_PROFILER_PATH_32</code>	<code>CONTRAST_CCC_COR_PROFILER_PATH_32</code>
<code>COR_PROFILER_PATH_64</code>	<code>CONTRAST_CCC_COR_PROFILER_PATH_64</code>

Then follow the usual setup instructions for your environment and application.

- **AppInsights in Azure App Service:**

As of version 20.9.3, the Contrast .NET Framework Site Extension now supports compatibility with Application Insights (using the CLR Instrumentation Engine (CIE)). There is no further action required to use AppInsights with the Contrast .NET Framework Site Extension. The .NET Framework agent's profiler will be loaded by the CIE if it is registered as the profiling tool in the Azure AppService instance (for example because AppInsights is enabled).

.NET Agent Explorer




IMPORTANT

Starting with .NET Core agent 4.0.0 and .NET Framework agent 51.0.0, Agent Explorer replaces the Contrast Tray application.

The .NET Agent Explorer is an application that displays high-level information about the health of the .NET Core and .NET Framework agents. Use this application to verify that the agent is working as expected, especially after you initially install the agent.

Installing an agent also installs this application. If you install both types of agents, only one instance of Agent Explorer is installed.

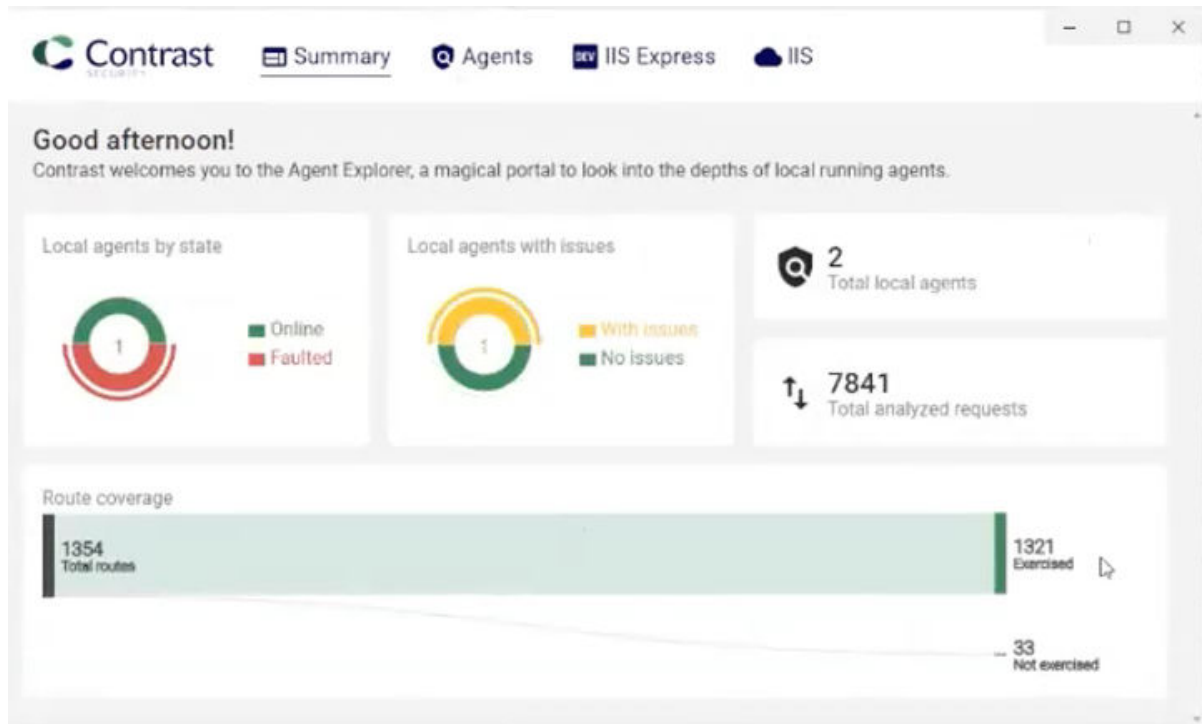
Agent Explorer access

After you install a .NET Core or .NET Framework agent, the Agent Explorer icon () displays in the tray. Right-click the icon to open the application.

Agent Explorer details

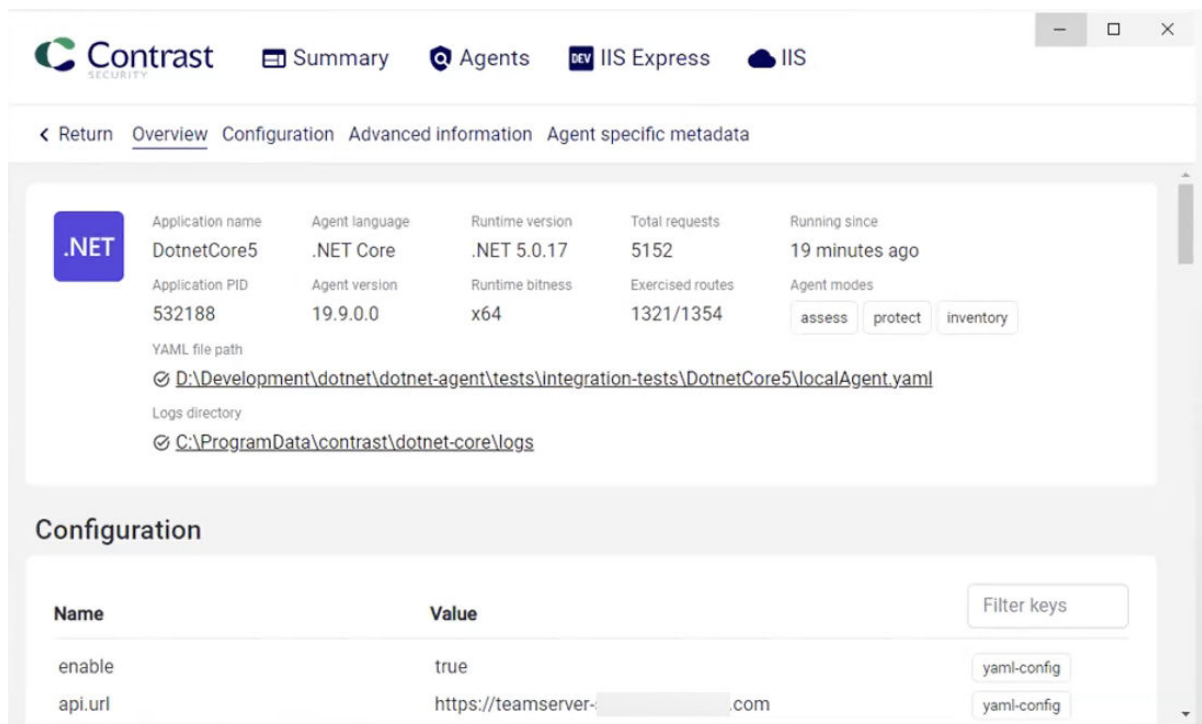
The Agent Explorer displays these details:

- **Summary**



This dashboard shows high level details about your agents, including their stage, whether any of them have issues, and route coverage.

- **Agents**



This tab provides details about the health of your .NET Core and .NET Framework agents. The Configuration section displays a message if Agent Explorer discovers a specific issue that is occurring.

You can access the agent's configuration (YAML) file directly from the link in the Overview section. Scroll down to see information about the agent's configuration, advanced information, and session metadata.

- **IIS Express**

This tab shows details for web applications running on IIS Express.

- **IIS**

This tab shows details for the web applications running on the IIS server.

Use application pools in IIS

The .NET agent automatically instruments all ASP.NET applications deployed to IIS. If you install the .NET agent and ensure the background Windows service runs for the agent, Contrast will instrument all IIS-hosted applications.

You might want to exclude some applications from instrumentation because:

- You don't need to gather security, architecture and library information for these applications.
- The applications are on resource-constrained servers or need to avoid additional performance demands for Contrast instrumentation.

Web applications hosted in IIS run in application pools. If you need to disable the .NET agent for an application, you can [denylist the application pool \(page 268\)](#) where it runs.

There are three ways to find the application pool that runs a specific application:

- **Internet Information Services (IIS) Manager**

Start IIS Manager with the command: `%windir%\system32\inetsrv\InetMgr.exe`. Select the web application you want, and select **Basic Settings**. You will see a field that displays the application pool name.

- **AppCmd.exe**

If you Administrator privileges, run `cmd.exe`. Navigate to `C:\Windows\System32\inetsrv`. Enter `appcmd list apps` to see a list of applications and the application pools for each.

- **Contrast .NET logs**

Start the Contrast .NET agent. Browse to an application. In Windows, navigate to `C:\ProgramData\Contrast\dotnet\LOGS`. Open the most recent Profiler log (`XXXXXX_Profiler_[AppDomain]XXXXXX[XX].log`). The application pool name is on the line that starts with **ApplicationPool Name**.

Denylist or allowlist an application pool

Denylists and allowlists are based on the application pool name. Application pool denylists and allowlists also accept `*` as a variable-length wildcard. (`AppPool*` will match `AppPool1`, `AppPool_arb`, etc.)

Allowlists take precedence over denylists. Application pools that satisfy both lists aren't analyzed.

To disable the agent for a specific application, populate `agent.dotnet.app_pool_denylist` with the appropriate application pool in `C:\ProgramData\Contrast\dotnet\contrast_security.yaml`:

```
# Comma-separated list of application pools ignored by Contrast
agent:
  dotnet:
    app_pool_denylist: ExampleAppPoolName
```

To only enable the agent for specific applications hosted by IIS, configure `agent.dotnet.app_pool_allowlist` to only analyze certain application pools. If an application pool is allowlisted, the agent analyzes the matching pools. There should be no performance impact for any other applications.

To enable the agent for only specific application pools, populate `agent.dotnet.app_pool_allowlist` with the appropriate application pool in `C:\ProgramData\Contrast\dotnet\contrast_security.yaml`:

```
# Comma-separated list of application pools exclusively profiled by Contrast agent:
dotnet:
  app_pool_allowlist: ExampleAppPoolName
```

**TIP**

Read more about [yaml configuration \(page 107\)](#).

.NET Framework and .NET Core Telemetry

.NET Framework and .NET Core agents use telemetry to collect usage data. Telemetry is collected when an instrumented application first loads the agent's sensors and then periodically (every few hours) afterwards.

[Your privacy is important to us \(page 1275\)](#). The telemetry feature doesn't collect application data. The data is anonymized before being sent securely to Contrast. Then the aggregated data is stored encrypted and under restricted access control. Any collected data will be deleted after one year.

The telemetry feature collects the following data:

Agent versions	Data	
.NET Framework later than 2020.8.3	Agent version	
.NET Core later than 1.5.15	Operating system and version	
	Whether the agent is running in a container	
	Whether the agent is running in Azure App Service	
	Hashed Media Access Control (MAC) address: a cryptographically (SHA256) anonymous and unique ID for a machine	
	Kernel version	
	Process running time	
	Whether Assess is enabled	
	Whether Protect is enabled	
	.NET Framework later than 2020.8.3	.NET Framework runtime version
	.NET Core later than 1.5.15	.NET Core runtime version
.NET Framework later than 20.9.1	Hosted or on-premises Contrast instance	
.NET Core later than 1.5.17		
.NET Framework later than 20.9.3	CLR Instrumentation Engine (CIE) usage	
.NET Core later than 1.5.19	Application framework	
	Chained profiler vendor	
.NET Framework later than 20.10.1	Process hosting mode	
.NET Core later than 1.5.20	CIE Raw Profiler Hook usage	
.NET Framework later than 20.10.2	Names of configuration settings with non-default values	
.NET Core later than 1.5.21	Names of disabled Assess rules	

Agent versions	Data
.NET Framework later than 20.12.2	Time elapsed for agent's profiler component to initialize
.NET Core later than 1.7.2	Time elapsed for agent's first request to the Contrast web interface
	Time elapsed for agent's profiler component to initialize
	Time elapsed between agent initialization and end of the first request
.NET Framework later than 21.1.1	Metrics on IIS-hosted applications, including: <ul style="list-style-type: none"> • Total application count • Application count that will be analyzed (pass application allow list/deny list configuration) • Count of apps hosted on CLR4 application pools • Count of apps hosted on CLR2 application pools Metrics on IIS applications pools <ul style="list-style-type: none"> • Total count • Count with agent attached • Count of CLR4 • Count of CLR2 Minimum number of applications in a single app pool
	Maximum number of applications in a single app pool
	Median number of applications across all app pools
.NET Framework later than 21.1.2	Rule mode (i.e. Monitor vs. Block) for each Protect rule
.NET Core later than 1.7.5	
.NET Framework later than 21.4.2	Exceptions thrown and caught within agent sensor code, including log message, exception type, exception message, and stack trace frames for <code>System</code> and <code>Contrast</code> methods.
.NET Core later than 1.8.4	
.NET Framework later than 21.7.1	• Process Architecture (x86/x64)
.NET Core later than 1.9.7	OS Architecture (x86/x64)
	In Azure App Service, the values of the following environment variables: <ul style="list-style-type: none"> • WEBSITE_PHYSICAL_MEMORY_MB • WEBSITE_PLATFORM_VERSION • WEBSITE_SKU
.NET Framework later than 21.9.2	Description of location where YAML config file was loaded from (i.e., path specified by environment variable, default location, application directory).
.NET Core later than 2.0.1	

To opt-out of the telemetry feature, set the `CONTRAST_AGENT_TELEMETRY_OPTOUT` environment variable to `1` or `true`.

Telemetry data is securely sent to telemetry.dotnet.contrastsecurity.com. You can also opt out of telemetry by blocking communication at the network level.

.NET Core agent

The Contrast .NET Core agent analyzes the behavior of .NET Core web applications as users interact with them.



NOTE

The latest .NET Core agent supports Assess (IAST), Protect (RASP), and SCA features.

The agent automatically instruments the ASP.NET Core application when the host process is set up with profiling environment variables or an application launch profile.

The Contrast .NET Core agent consists of two components that run within the same process as your application:

- The **.NET Profiler** instruments applications by adding calls to Contrast sensor code in security relevant APIs used by the application and its dependencies (also known as IL weaving).
- **Sensors** gather security, architecture and library information.

Once you [install the .NET Core agent \(page 273\)](#), its sensors will gather information about the application's security, architecture and libraries as users exercise the applications. You can view the results of the agent's analysis in Contrast. The agent uses these [supported technologies \(page 271\)](#) and these [system requirements \(page 272\)](#).

.NET Core supported technologies

We support the following technologies for this agent.

Technology	Supported versions	Notes
Application frameworks	<ul style="list-style-type: none"> • ASP.NET Core (3.1.X, 5.0.X, 6.0.X, 7.0.X, 8.0X) • Model-View-Controller (MVC) • Razor Pages • Grpc.AspNetCore • Blazor (6.0.X, 7.0.X, 8.0X) • Blazor Server (Server-Side) only, Blazor WebAssembly is not supported • SignalR (6.0.X, 7.0.X, 8.0X) <ul style="list-style-type: none"> • Transport protocols: WebSockets, Server-Sent Events, and Long Polling 	<p>Limited support</p> <p>ASP .NET Core 3.1.X and 5.0.X</p> <p>Not supported:</p> <ul style="list-style-type: none"> • .NET Core or ASP.NET Core version 2.1 or below • ASP.NET Core applications running under the .NET Framework (Windows) or Mono (Linux/Windows) • Grpc.Core The future of gRPC in C# belongs to grpc-dotnet contains additional details.
Runtime	<ul style="list-style-type: none"> • .NET Core Runtimes: 3.1.X, 5.0.X, 6.0.X, 7.0.X, 8.0X, 9.0.X • .NET Core target framework monikers: <ul style="list-style-type: none"> • netcoreapp3.1 • net5.0 • net6.0 • net7.0 • net8.0 • net9.0 	<p>Limited support:</p> <p>.NET Core Runtimes 3.1.x and 5.0.X</p> <p>Not supported:</p> <ul style="list-style-type: none"> • Running with an ASP.NET Core application that's a higher version than the runtime (for example, an application with the .NET Core 3.1 runtime that references ASP.NET Core 5.0) • Running with a .NET Core application for which the referenced ASP.NET Core version and the target runtime selected during compilation time don't match
.NET Core for Windows		
Windows operating systems	<ul style="list-style-type: none"> • Windows Server (LTSC) (x86, x64): 2012 R2, 2016, 2019, 2022 • Windows Server (SAC) (x64): 1809, 1903 • Windows workstation (x86, x64): 7, 8/8.1, 10 	<p>On 64-bit systems, you can use the agent to analyze both 32-bit and 64-bit web applications.</p> <p>Not supported:</p> <ul style="list-style-type: none"> • Windows on ARM
Server container	Kestrel, IISHttpServer	<p>Not supported:</p> <p>Http.sys (formerly called WebListener)</p>
Hosting container	Self-hosted, IIS, IIS Express	
.NET Core for Linux operating systems		

Technology	Supported versions	Notes
Linux operating systems	<ul style="list-style-type: none">• Ubuntu: 18.04 and later (x64, ARM64)• Debian: 10 and later (x64, ARM64)• openSUSE: 15 and later (x64)• Alpine: 3.13 and later (x64, ARM64)• CentOS Stream 8 and later (x64)• Red Hat Enterprise Linux: 7 and later (x64)	Not supported: Red Hat Enterprise Linux 6
Server container	Kestrel	
Hosting container	Self-hosted	



IMPORTANT

- .NET Core 2.2 is not supported after the .NET core agent version 1.5.20. If you are using .NET Core 2.2, you'll need to use .NET Core agent version 1.5.20 or lower until you can upgrade your application's .NET Core runtime.
- As of .NET core agent version 1.9.9, we no longer support .NET Core 2.1. If you are using .NET Core 2.1, you'll need to use .NET Core agent version 1.9.9 or lower until you can upgrade your application's .NET Core runtime.
- Microsoft support for .NET 5.0 ended on May 10th, 2022 and support for .NET Core 3.1 ended on December 13th, 2022. Contrast support for .NET 5.0 and .NET Core 3.1 entered limited support with .NET Core agent version 3.0.0. Under limited support, Contrast will only solve problems that can be reproduced under supported language versions. Contrast strongly recommends that you upgrade your applications to a supported version of .NET.



NOTE

The .NET Core agent does not support applications that do not reference `System.Runtime` and ASPNET Core. The agent also does not support **trimmed self-contained deployments and executables**, because the compiler can potentially trim assemblies that the agent depends on.

.NET system requirements

Before installing the .NET agent, confirm you can meet the following requirements:

- You have administrative access to a server, and the **server is supported by Contrast (page 271)**.
- There is a deployed application to be analyzed, and the **web application technology is supported (page 271)** by Contrast.
- The web server has network connectivity with Contrast.
- The server meets the minimum requirements (stated below).

Requirement	Recommended	Minimum	Notes
CPU	at least 4	2	

Requirement	Recommended	Minimum	Notes
Memory	at least 8 GB	4 GB	Agents running in Assess roughly double the memory requirements of analyzed applications. Applications should use less than half of the available memory when an agent isn't installed.
Operating system	<ul style="list-style-type: none">WindowsLinux		macOS is not supported.
Processor architecture	<ul style="list-style-type: none">32-bit x86 processors64-bit x86 processors64-bit ARM processors		Windows on ARM processors is not supported.

Install the .NET Core agent

To install the .NET Core agent:

1. Place the agent's components on the server's file system.
2. Set environment variables so that the .NET runtime loads the agent's profiler component.
3. Use the application as you normally would and verify that Contrast sees your application.

Depending on your situation, use one of these installation methods:

- [Manual installation \(page 273\)](#) (if you are using self-hosted web application running on Windows, Linux or Docker)
- [.NET Core agent for IIS installer \(page 281\)](#) (if you are using IIS)
- [Azure App Service \(page 278\)](#)
- [NuGet \(page 277\)](#)

To auto-upgrade your agent, enable this option with the [Agent Upgrade Service \(page 226\)](#).

Install the .NET Core agent manually

Use this method to install the .NET Core agent if you are using a web application hosted on IIS, or running a self-hosted application on Windows, Linux or Docker.



NOTE

Installing within containers can be complex, and these steps might not work for your situation. Read more about [installing with Docker](#).

Before you begin

Check the [system requirements \(page 272\)](#) and [supported technologies \(page 271\)](#) to be sure installation will work and ensure best performance.

Steps

1. Download the agent:
 - a. At the top right of the Contrast web interface, select **Add new**.
 - b. Select the **Application** card.
 - c. Select the operating system you are using.
 - d. Select a method to install the agent.
 - e. Download the agent configuration file.

**TIP**

Use the [Contrast agent configuration editor \(page 109\)](#) to make the agent configuration easier.

- f. Download the agent.
2. On the web server, extract the downloaded ZIP archive (for example, *Contrast.NET.Core_1.0.1.zip*) to a directory that your applications have sufficient permissions to access.
3. Set the following environment variables on your application's process. Use the appropriate CORECLR_PROFILER_PATH settings for your operating system. Replace <UnzippedDirectoryRoot> with your archive directory.

- **Windows**

Environment variable	Value
CORECLR_PROFILER_PATH_64	<UnzippedDirectoryRoot>\runtimes\win-x64\native\ContrastProfiler.dll
CORECLR_PROFILER_PATH_32	<UnzippedDirectoryRoot>\runtimes\win-x86\native\ContrastProfiler.dll
CORECLR_PROFILER	{8B2CE134-0948-48CA-A4B2-80DDAD9F5791}
CORECLR_ENABLE_PROFILING	1
CONTRAST_CONFIG_PATH	<path_to_contrast_security.yaml>

**IMPORTANT**

If you are running the .NET Core agent and the .NET Framework agent on the same server, the CONTRAST_CONFIG_PATH option applies to the [load path \(page 106\)](#) for both agents. To apply distinct paths for each agent, use these options to set the data directory:

- CONTRAST_CORECLR_DATA_DIRECTORY
- CONTRAST_DATA_DIRECTORY

- **Linux x64**

Environment variable	Value
CORECLR_PROFILER_PATH_64	<UnzippedDirectoryRoot>/runtimes/linux-x64/native/ContrastProfiler.so
CORECLR_PROFILER	{8B2CE134-0948-48CA-A4B2-80DDAD9F5791}
CORECLR_ENABLE_PROFILING	1
CONTRAST_CONFIG_PATH	<path_to_contrast_security.yaml>

- **Linux ARM64**

Environment variable	Value
CORECLR_PROFILER_PATH_64	<UnzippedDirectoryRoot>/runtimes/linux-arm64/native/ContrastProfiler.so
CORECLR_PROFILER	{8B2CE134-0948-48CA-A4B2-80DDAD9F5791}
CORECLR_ENABLE_PROFILING	1
CONTRAST_CONFIG_PATH	<path_to_contrast_security.yaml>

4. Ensure the following paths are accessible by the runtime user of the application.

Path	Usage	Customizable	Permissions
The path to .NET Core YAML (page 292)	Configures the agent	Yes; set the environment variable <code>CONTRAST_CONFIG_PATH</code>	Read
<code><UnzippedDirectoryRoot></code>	The root "installation" directory; stores the agent binaries	No	Read
<ul style="list-style-type: none">• Windows: <code>%ProgramData%\Contrast\dotnet-core\logs</code>• Linux: <code>/var/tmp/contrast/dotnet-core/logs</code>	Directory for Contrast agent logs. If missing, the directory will be created	Yes; set the environment variable <code>CONTRAST_CORECLR_LOGS_DIRECTORY</code>	Read/Write(or inherited from a parent directory)



NOTE

When running in IIS, make sure that the application pool can access these paths.

For example, given an application pool called `Default Web Site` using the default identity `ApplicationPoolIdentity`, ensure that the user `IIS AppPool\Default Web Site` has effective permissions to read the unzipped directory root.

5. [Configure the agent \(page 290\)](#) with authentication credentials and proxy settings to connect to Contrast.
6. Once the application has loaded, use the application and then verify that the server and application are active in Contrast, and that any expected vulnerabilities appear.



TIP

To [update the agent \(page 288\)](#), replace the agent files in the agent directory and restart your application. As the agent is running alongside your application, it can't update itself.

The agent automatically starts with your application as long as the environment is properly set up.

To stop the agent, stop the application and remove agent from its environment. Alternatively, you may change the `CORECLR_ENABLE_PROFILING` setting to 0.

Follow any of these examples to set environment variables using:

- [IIS \(page 275\)](#)
- [Bash \(Linux\) \(page 276\)](#)
- [Powershell or Powershell Core \(Windows\) \(page 276\)](#)
- [Launch profile \(dotnet.exe\) \(page 277\)](#)

IIS and IIS Express

Set the environment variables with either:

- [The `environmentVariables` section in the application `web.config`](#)

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <system.webServer>
```

```
<!-- ... -->
<aspNetCore processPath="dotnet"
arguments=".\\ExampleNetCoreApp.dll" stdoutLogEnabled="false"
stdoutLogFile=".\\logs\\stdout">
  <environmentVariables>
    <environmentVariable name="CORECLR_PROFILER_PATH_64"
value="C:\\contrast\\dotnetcore\\runtimes\\win-
x64\\native\\ContrastProfiler.dll" />
    <environmentVariable name="CORECLR_PROFILER_PATH_32"
value="C:\\contrast\\dotnetcore\\runtimes\\win-
x86\\native\\ContrastProfiler.dll" />
    <environmentVariable name="CORECLR_ENABLE_PROFILING"
value="1" />
    <environmentVariable name="CORECLR_PROFILER"
value="{8B2CE134-0948-48CA-A4B2-80DDAD9F5791}" />
    <environmentVariable name="CONTRAST_CONFIG_PATH"
value="C:\\contrast\\dotnet-core\\contrast_security.yaml" />
  </environmentVariables>
</aspNetCore>
</system.webServer>
</configuration>
```

- The [application pool setting](#) on the server

Bash (Linux)

Linux x64:

```
export CORECLR_PROFILER_PATH_64=/usr/local/contrast/runtimes/linux-x64/
native/ContrastProfiler.so
export CORECLR_ENABLE_PROFILING=1
export CORECLR_PROFILER={8B2CE134-0948-48CA-A4B2-80DDAD9F5791}
export CONTRAST_CONFIG_PATH=/etc/contrast/contrast_security.yaml
```

Linux ARM64:

```
export CORECLR_PROFILER_PATH_64=/usr/local/contrast/runtimes/linux-arm64/
native/ContrastProfiler.so
export CORECLR_ENABLE_PROFILING=1
export CORECLR_PROFILER={8B2CE134-0948-48CA-A4B2-80DDAD9F5791}
export CONTRAST_CONFIG_PATH=/etc/contrast/contrast_security.yaml
```

Then run the application:

```
dotnet ./MyAppWithContrastAgent.dll
```

Powershell or Powershell Core (Windows)

```
$env:CORECLR_PROFILER_PATH_64 = 'C:\\contrast\\dotnetcore\\runtimes\\win-
x64\\native\\ContrastProfiler.dll'
$env:CORECLR_PROFILER_PATH_32 = 'C:\\contrast\\dotnetcore\\runtimes\\win-
x86\\native\\ContrastProfiler.dll'
$env:CORECLR_ENABLE_PROFILING = '1'
$env:CORECLR_PROFILER = '{8B2CE134-0948-48CA-A4B2-80DDAD9F5791}'
$env:CONTRAST_CONFIG_PATH = 'C:\\contrast\\dotnet-core\\contrast_security.yaml'
```

Then run the application:

```
dotnet .\\MyAppWithContrastAgent.dll
```


Launch profile (dotnet.exe)

```
{
  "MyAppWithContrastAgent": {
    "environmentVariables": {
      "CORECLR_PROFILER_PATH_64": "C:\\\\contrast\\\\dotnetcore\\\\runtimes\\
\\win-x64\\\\native\\\\ContrastProfiler.dll",
      "CORECLR_PROFILER_PATH_32": "C:\\\\contrast\\\\dotnetcore\\\\runtimes\\
\\win-x86\\\\native\\\\ContrastProfiler.dll",
      "CORECLR_ENABLE_PROFILING": "1",
      "CORECLR_PROFILER": "{8B2CE134-0948-48CA-A4B2-80DDAD9F5791}",
      "CONTRAST_CONFIG_PATH": "c:\\\\contrast\\\\config\\\\MyApp\\
\\contrast_security.yaml"
    }
  }
}
```

Then run the application:

```
dotnet run --launch-profile MyAppWithContrastAgent
```

Install the .NET Core agent manually with NuGet

In some instances, you may prefer to manually install the .NET Core agent using NuGet. For example, this can be useful if you are unable to access the [Azure App Service site extension \(page 278\)](#) or if you prefer to include the .NET Core agent as a dependency.

Before you begin

[Single-file deployments](#) are not supported when you install the .NET Core agent manually with NuGet.



IMPORTANT

When redeploying a web application that has Contrast agent running, you may run into an error that says "Files in use" on *ContrastProfiler.dll*. This happens because the agent DLL files are locked by .NET, and can't be overwritten while the application is still running.

Steps

1. Add the Contrast NuGet package to your application.
Using dotnet command line:

```
dotnet add package Contrast.SensorsNetCore
```

Using Visual Studio:

- Under the application project in the Solution Explorer, right-click on **References** and select **Manage NuGet Packages**.
 - Search for the **Contrast.SensorsNetCore** package, select it and add it to your project.
 - Build your application. Confirm that a *contrast* folder appears in your project. When the application is published, this folder also appears in the build output directory.
2. Set environment variables so that the .NET runtime loads the agent's profiler component.:
Windows:

```
CORECLR_ENABLE_PROFILING: 1
CORECLR_PROFILER: {8B2CE134-0948-48CA-A4B2-80DDAD9F5791}
CORECLR_PROFILER_PATH_32: <application directory>\contrast\runtimes\win-
x86\native\ContrastProfiler.dll
CORECLR_PROFILER_PATH_64: <application directory>\contrast\runtimes\win-
x64\native\ContrastProfiler.dll
```

Linux x64:

```
CORECLR_ENABLE_PROFILING: 1
CORECLR_PROFILER: {8B2CE134-0948-48CA-A4B2-80DDAD9F5791}
CORECLR_PROFILER_PATH: <application directory>/contrast/runtimes/linux-
x64/native/ContrastProfiler.so
```

Linux ARM64:

```
CORECLR_ENABLE_PROFILING: 1
CORECLR_PROFILER: {8B2CE134-0948-48CA-A4B2-80DDAD9F5791}
CORECLR_PROFILER_PATH: <application directory>/contrast/runtimes/linux-
arm64/native/ContrastProfiler.so
```

3. Set the basic configuration either with the [YAML configuration file \(page 292\)](#) or with [environment variables \(page 291\)](#). For example:

```
CONTRAST_CONFIG_PATH: [Path to yaml config file]
```

At minimum, the following environment variables are required:

```
CONTRAST__API__URL: [IF USING ANOTHER SERVER THAN THE DEFAULT: https://
app.contrastsecurity.com]
CONTRAST__API__USER_NAME: [REPLACE WITH YOUR AGENT USERNAME]
CONTRAST__API__SERVICE_KEY: [REPLACE WITH YOUR AGENT SERVICE KEY]
CONTRAST__API__API_KEY: [REPLACE WITH YOUR AGENT API KEY]
```

4. Deploy your application with the environment variables from the previous step.
5. Once the application has loaded, use the application and then verify that the server and application are active in Contrast, and that any expected vulnerabilities appear.

Install the .NET Core agent with Azure App Service

Use this procedure for an express installation of the .NET Core agent using Azure Portal Extensions.

Before you begin

Before you begin, check the [system requirements \(page 272\)](#) and [supported technologies \(page 271\)](#) to be sure installation will work and ensure best performance.

Steps

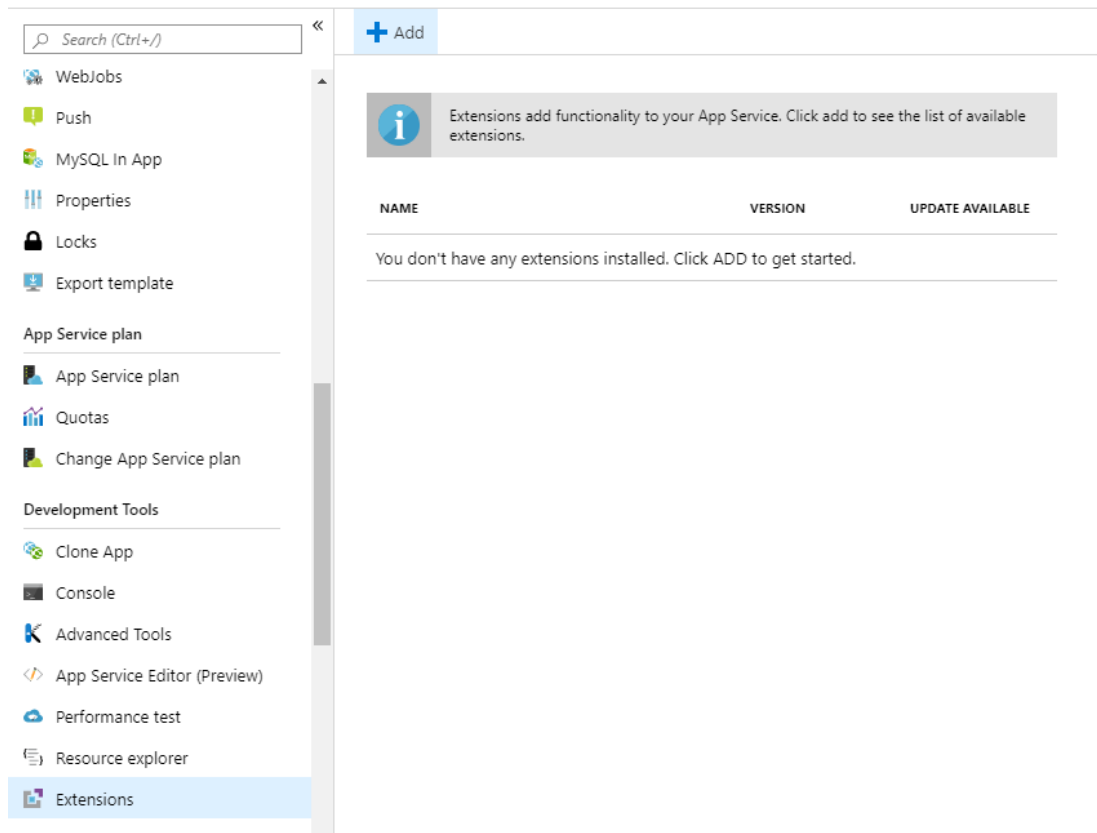
1. Create an [Azure account](#), if you don't have one already.
2. Create a [.NET web application](#) and deploy it to Azure App Service.
3. Publish your application to Azure, and confirm that it works as expected without Contrast.
4. Ensure that your application is deployed using a Windows plan. (Linux plans do not support Site Extensions.)

**NOTE**

If you do not have access to the site extension, you can [install the .NET Core agent manually with NuGet \(page 277\)](#).

5. Add the Contrast .NET Core Site Extension:

- With the Azure portal
 - a. In the Azure Portal, select your hosted application.
 - b. Select **Extensions**.



- c. Select **Add**.
- d. Select the **Contrast .NET Core Site Extension for Azure App Service**. This is the extension for .NET Core applications.
- e. Select **OK**, and agree to the terms and conditions.
- f. Wait a few seconds and confirm the site extension installed correctly.

**NOTE**

The site extension sets a number of environment variables, including:

```
CORECLR_ENABLE_PROFILING=1
CORECLR_PROFILER={8B2CE134-0948-48CA-A4B2-80DDAD9F5791}
CORECLR_PROFILER_PATH_32=D:\Home\siteextensions\Contrast.NetCore.Azure.SiteExtension\ContrastNetCoreAppService\runtimes\win-x86\native\ContrastProfiler.dll
CORECLR_PROFILER_PATH_64=D:\Home\siteextensions\Contrast.NetCore.Azure.SiteExtension\ContrastNetCoreAppService\runtimes\win-x64\native\ContrastProfiler.dll
CONTRAST_INSTALL_DIRECTORY=D:\Home\siteextensions\Contrast.NetCore.Azure.SiteExtension\ContrastNetCoreAppService\MicrosoftInstrumentationEngine_ConfigPath32_ContrastCoreX86
Config=D:\Home\siteextensions\Contrast.NetCore.Azure.SiteExtension\ContrastCieCoreClrProfiler-32.config
MicrosoftInstrumentationEngine_ConfigPath64_ContrastCoreX64
Config=D:\Home\siteextensions\Contrast.NetCore.Azure.SiteExtension\ContrastCieCoreClrProfiler-64.config
```

If the CLR instrumentation engine (CIE) is configured for the application (for example, because Application Insights is enabled), Azure should automatically overwrite the `CORECLR_PROFILER*` variables to point to the profiler of the CIE.

The CIE will then use the `MicrosoftInstrumentationEngine_*` variables to load the Contrast agent.

If the CIE is not configured for the application, the standard `CORECLR_PROFILER*` variables will be used to load the Contrast agent.

- With the Azure CLI
 - Use a command similar to this one for a .NET Core Site Extension:

```
az resource create --resource-group 'myResourceGroup' --resource-type Microsoft.Web/sites/siteextensions --name myAppService/siteextensions/Contrast.NetCore.Azure.SiteExtension --properties "{ }
```

In this example, the command adds a Contrast .NET Core Site Extension to an App Service named "myAppService" in Resource Group "myResourceGroup"

After you add the extension, the Azure Portal displays a list of the installed agents with details similar to the following:

Name	Version	Update Available
Contrast.NET Core Site Extension for Azure App Service	4.2.4	No

**TIP**

You can also install the agent from the Site Extensions area of your application management SCM (Kudu) site.

**IMPORTANT**

If a new version of the .NET Core agent is available, it's indicated in the Azure Portal or Kudu dashboard. You must stop the site before starting the update; otherwise, the update may fail.

6. Add configuration options

- With the Azure Portal
 - a. In the Azure Portal, select your hosted application.
 - b. Select **Configuration** under **Settings** to configure settings that allow the agent to connect to Contrast.
 - c. Select **New application setting** and add the following values for your application:

Key	Value
CONTRAST__API__USER_NAME	Replace with your agent username (page 104) .
CONTRAST__API__SERVICE_KEY	Replace with your agent service key (page 104) .
CONTRAST__API__API_KEY	Replace with your agent API key (page 104) .
CONTRAST__API__URL	Defaults to https://app.contrastsecurity.com . Replace with another URL, if you're using a Contrast application that's hosted elsewhere.

- With the Azure CLI
 - Enter a command similar to this one:

```
az webapp config appsettings set --resource-group 'myResourceGroup'
--name 'myAppService' --settings CONTRAST__API__URL=https://
app.contrastsecurity.com CONTRAST__API__API_KEY={Your API
KEY} CONTRAST__API__SERVICE_KEY={Your Service key}
CONTRAST__API__USER_NAME={Your agent user}
```

Get API values ([agent keys \(page 104\)](#)) from the Contrast web interface or by downloading a YAML file for the .NET Core agent.

7. In the Azure Portal, go to the application overview and **Restart** the application.
Running the application automatically instruments any application that is running inside of the App Service. You should begin to see data in Contrast
8. Navigate to the application and confirm the application is reporting to Contrast.
You can view log files to verify that Contrast is running:
 - a. In the Azure Portal, go to **Advanced Tools** for the app service.
 - b. Select **Go**.
 - c. In the Kudu Services window, select “Debug console” menu at the top and select “CMD”.
 - d. Select the `LogFiles` directory.
 - e. Select the `Contrast` directory.
 - f. Select the `dotnet` directory.
You will see an agent log named `<PID>_Profiler_<App Service Name>_<XXX>.log`.
 - g. Verify that there are no ERROR log entries.

Install the .NET Core agent with the .NET Core agent for IIS installer

The .NET Core agent for IIS installer is a normal Windows application installer built using standard MSI technology. It validates that the target server and satisfies several requirements (for example, that the server's operating system is a supported operating system). If all requirements are met, the installer:


- Registers the .NET Core agent for IIS as a standard Windows program.
- Places the agent's files on a disk in the specified install location (for example, `C:\Program Files\Contrast\dotnet-core`). This includes several dynamic link libraries (DLLs) and executables.

- Creates the specified data directory for the agent that's primarily used to store agent log files and configuration (for example, `C:\ProgramData\Contrast\dotnet-core`).
- Adds the .NET Core agent's native modules to IIS.

Before you begin

Before you begin, check the [system requirements \(page 272\)](#) and [supported technologies \(page 271\)](#) to be sure installation will work and ensure best performance.

Install the agent using Contrast

1. In the Contrast web application, select **Add new**.
2. Choose .NET Core in the **Choose an agent** dropdown menu.
3. Select the link  **.NET Core IIS agent installer** under **Install with IIS**. A ZIP archive downloads.
4. Extract the downloaded ZIP archive on the web server, and run `contrast-dotnet-core-agent-for-iis-installer.exe`. This installs the .NET Core agent for IIS.



TIP

You can use the command line to access additional options supported by the .NET Core agent for IIS installer.

5. [Configure the .NET Core agent with a YAML configuration file \(page 292\)](#) to set the [authentication keys \(page 104\)](#) and any application-specific configuration.
6. Copy the yaml file to `C:\ProgramData\Contrast\dotnet-core` if not already there.
7. Restart IIS to pick up the changes.
8. Use the application as you normally would and verify that Contrast sees your application.

Install the agent using command line

Use the command line to access additional options supported by the .NET Core agent for IIS installer.

The .NET Core for IIS agent can be installed using the Windows interface, and uninstalled or repaired using standard Windows features (including the Programs and Features Control Panel and Powershell). However, you may want to use the Contrast Windows installer to perform these actions instead for certain scenarios such as automated scripting.


Use these commands for attended mode:

- **Install:**`contrast-dotnet-core-agent-for-iis-installer.exe`
- **Uninstall:**`contrast-dotnet-core-agent-for-iis-installer.exe -uninstall`
- **Repair:**`contrast-dotnet-core-agent-for-iis-installer.exe -repair`

Use these commands for unattended or silent mode:

- **Install:**`contrast-dotnet-core-agent-for-iis-installer.exe -s SUPPRESS_RESTARTING_IIS=1`
- **Uninstall:**`contrast-dotnet-core-agent-for-iis-installer.exe -uninstall -s SUPPRESS_RESTARTING_IIS=1`
- **Repair:**`contrast-dotnet-core-agent-for-iis-installer.exe -repair -s SUPPRESS_RESTARTING_IIS=1`

The .NET Core agent for IIS installer supports several additional options that are accessible when you use the command line for installation.

Option	Description	Example
INSTALLFOLDER	This option specifies the install directory for the agent files.	INSTALLFOLDER=C:\Program Files\Contrast\dotnet-core
AGENT_EXPLORER_INSTALLFOLDER	This option specifies the directory for Agent Explorer files.	AGENT_EXPLORER_INSTALLFOLDER="C:\Program Files\Contrast\agent-explorer"
INSTALL_AGENT_EXPLORER	If you don't want to install the Agent Explorer, set the value for this option to 0. The default value is 1, which installs the Agent Explorer.	INSTALL_AGENT_EXPLORER=1
DATAFOLDER	This option specifies the default location for agent log and configuration files.	DATAFOLDER=C:\ProgramData\Contrast\dotnet-core
SUPPRESS_RESTARTING_IIS	If you set the value of this option to 1, the installer does not restart IIS. The default value is 0.	SUPPRESS_RESTARTING_IIS=0
<div>  NOTE <ul style="list-style-type: none"> Applications do not load the agent until IIS restarts. Setting <code>SUPPRESS_RESTARTING_IIS</code> will prevent an auto-upgrade from running unless IIS does not have any active workers when the upgrade runs. </div>		
SKIP_IIS_MODULES	If you don't want to install the agent IIS modules, set the value of this option to 1. The default value is 0, which installs the IIS modules.	SKIP_IIS_MODULES=1
INSTALL_UPGRADE_SERVICE	If you don't want to install the agent upgrade service, set the value of this option to 0. The default value is 1, which installs the agent upgrade service.	INSTALL_UPGRADE_SERVICE=1
UPGRADE_SERVICE_INSTALLFOLDER	This option specifies the directory for the upgrade service files.	UPGRADE_SERVICE_INSTALLFOLDER="C:\Program Files (x86)\Contrast\upgrade-service"



IMPORTANT

The .NET Core agent for IIS installer automatically restarts IIS when you install the agent for the first time. You may want to change the configuration of any web server monitoring tools that raise alarms when IIS restarts.

The .NET Profiling API requires that profiled processes be started with a profiler. Therefore, the .NET Core agent must restart IIS (and any IIS worker processes) to attach the Contrast profiler. This process is similar to how other profiling products (for example, memory or performance profilers) behave.

Install the .NET Core agent in a container

Before you begin

- This topic provides general guidance for installing the Contrast .NET Core agent in a containerized application, with Docker as an example.
- You should have a basic understanding of how containers and related software work. You may need to adjust the instructions to meet your specific circumstances.
- If you are using Kubernetes, consider using the [Agent Operator \(page 552\)](#) to configure the agent.

Step 1: Install the agent

Contrast can be added either before or after the application is added to the container image. The recommended approach is with the use of named [multi-stage builds](#). For example:

```
FROM mcr.microsoft.com/dotnet/aspnet:6.0

# Hidden for brevity...

# Copy the required agent files from the official Contrast agent image.
COPY --from=contrast/agent-dotnet-core:latest /contrast /contrast
```

Where in this example, the latest .NET Core agent is used (check DockerHub for available tags).

Step 2: Configure the agent

Contrast agents accept configuration from multiple sources, with order of precedence documented in the [order of precedence \(page 106\)](#) section.

A mixed approach is recommended:

- Use a YAML file so that common configuration may be shared between many applications.
- Use environment variables for application-specific configuration values, to override values provided by a YAML file, or for sensitive keys that are injected during runtime.

YAML file configuration:

When using a [YAML file to configure the agent \(page 107\)](#), the environment variable `CONTRAST_CONFIG_PATH` can also be used to indicate where the YAML file is located inside the container.

For example, given a YAML file called `contrast_security.yaml` that exists in the Docker build context:

The environment variable `CONTRAST_CONFIG_PATH` can also be used to indicate where the YAML file is located.

```
agent:
  logger:
    path: /var/tmp
    level: WARN
```

The YAML file can be added to the container image as follows:

```
FROM mcr.microsoft.com/dotnet/aspnet:6.0

# Hidden for brevity...

# Add the Contrast agent to the image.
COPY --from=contrast/agent-dotnet-core:latest /contrast /contrast
```




```
# Copy the contrast_security.yaml file from Docker build context.
COPY ./contrast_security.yaml /contrast_security.yaml

# Finally configure the agent to use the YAML file previously copied.
ENV CONTRAST_CONFIG_PATH=/contrast_security.yaml
```

Environment variable configuration:

To set an application-specific configuration, use [environment variables \(page 110\)](#). Below are some common configuration options.

Title	Usage	Environment variable
Application name	Specify the application name reported to Contrast.	CONTRAST__APPLICATION__NAME
Application group	Specify the application access group for this application during onboarding.	CONTRAST__APPLICATION__GROUP
<div>  NOTE Application access groups have to be created first in Contrast. </div>		
Application tags	Add labels to an application.	CONTRAST__APPLICATION__TAGS
Server name	Specify the server name reported to Contrast.	CONTRAST__SERVER__NAME
Server environment	Specify in which environment the application is running. Valid values for this configuration are: Development, QA and Production	CONTRAST__SERVER__ENVIRONMENT
Server tag	Add labels to the server.	CONTRAST__SERVER__TAG

Step 3: Add profiler variables and authentication credentials

To enable instrumentation of your application, the .NET agent requires [additional environment variables \(page 273\)](#). The `CORECLR_` variables load the agent and the `CONTRAST_` variables are for agent authentication to the server.

Using the Dockerfile example from before:

x64

```
FROM mcr.microsoft.com/dotnet/aspnet:6.0

# Hidden for brevity...

COPY --from=contrast/agent-dotnet-core:latest /contrast /contrast

# Required variables to load the agent.
ENV CORECLR_PROFILER_PATH_64=/contrast/runtimes/linux-x64/native/ContrastProfiler.so \
    CORECLR_ENABLE_PROFILING=1 \
    CORECLR_PROFILER={8B2CE134-0948-48CA-A4B2-80DDAD9F5791}
```

ARM64

```
FROM mcr.microsoft.com/dotnet/aspnet:6.0

# Hidden for brevity...

COPY --from=contrast/agent-dotnet-core:latest /contrast /contrast
```

```
# Required variables to load the agent.
ENV CORECLR_PROFILER_PATH_64=/contrast/runtimes/linux-arm64

CORECLR_ENABLE_PROFILING=1 \
CORECLR_PROFILER={8B2CE134-0948-48CA-A4B2-80DDAD9F5791}

/native/ContrastProfiler.so \
```

Additionally, the [following environment variables \(page 104\)](#) are required for agent authentication to the server.

```
CONTRAST__API__URL=https://app.contrastsecurity.com/Contrast
CONTRAST__API__API_KEY={Your API KEY here}
CONTRAST__API__SERVICE_KEY={Your Service key here}
CONTRAST__API__USER_NAME={Your agent username here}
```

You can get API values ([agent keys \(page 104\)](#)) from Contrast or by downloading a YAML file for the .NET Core agent.



IMPORTANT

The `API_KEY`, `SERVICE_KEY` and `USER_NAME` keys should be considered sensitive data and handled accordingly. Contrast recommends injecting these during runtime from your secrets store (e.g. Kubernetes Secrets).

Step 4: Instrument your application

You can now run the application image with Contrast enabled. Contrast will instrument your application during startup and begin reporting security vulnerabilities to Contrast. You can verify that Contrast is running by checking the container.

Agent upgrade service

The Agent Upgrade Service is a background Windows service that helps you keep the .NET Framework and .NET Core for IIS agents automatically updated to the most recent version on Windows. The Agent Upgrade Service is included with the .NET Framework Agent Installer and .NET Core Agent for IIS Installer; the agent installers install two products:

- the corresponding agent, and
- the Agent Upgrade Service.

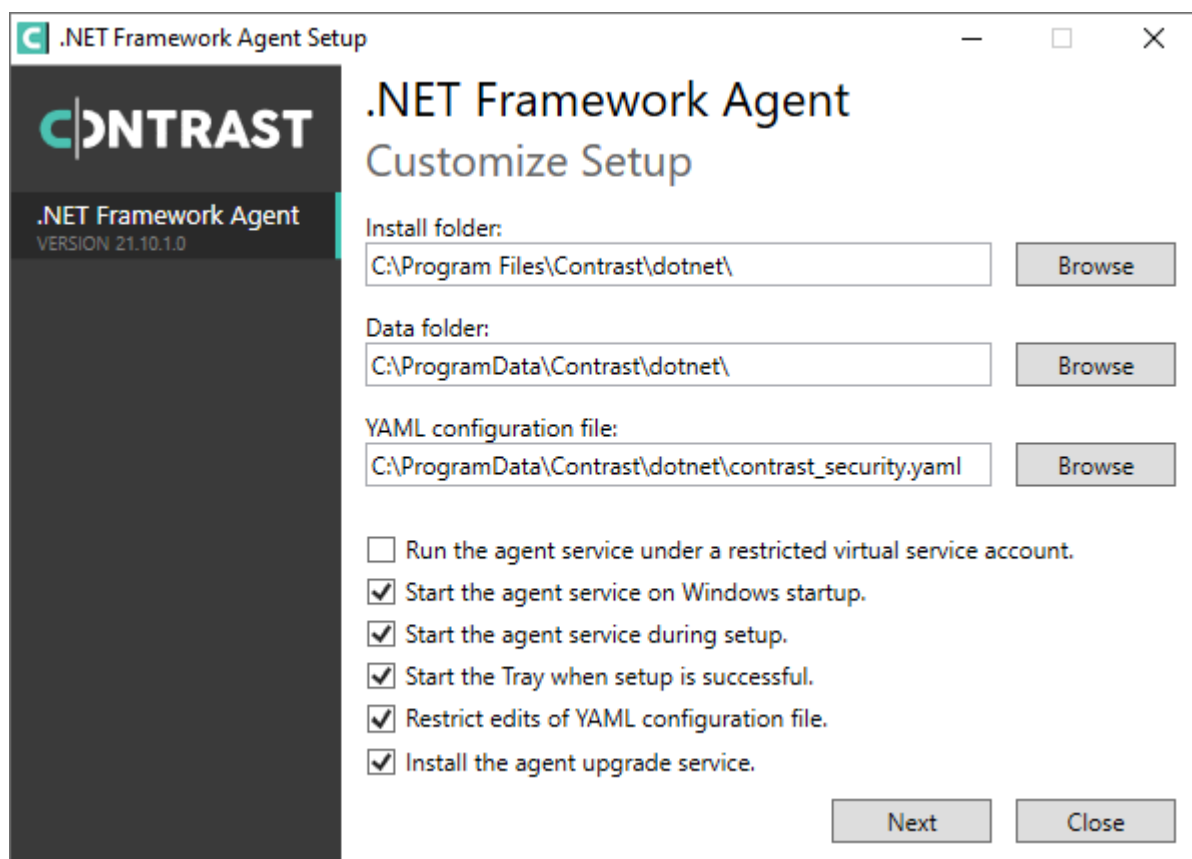
By default, the Agent Upgrade Service checks for new agent versions released to NuGet when the service first starts up (when the Windows Server is restarted.) If a new agent version is found, the Upgrade Service will download the new agent version, verify the installer's signature, and then finally execute the installer.

**NOTE**

When a new agent version installed, IIS will be restarted.

The Agent Upgrade Service is an optional component and is not required for agent Assess and Protect features.

- De-select the **Install the agent upgrade service** checkbox when installing the agent if you do not want to use the Agent Upgrade Service.
- If installing the agent via command line, add `INSTALL_UPGRADE_SERVICE=0` argument to not install the Agent Upgrade Service.



The behavior of the Agent Upgrade Service can be modified via an agent-specific configuration file in the Contrast data directory. The default location is `C:\ProgramData\Contrast\upgrade-service`.

The configuration for upgrading the .NET Core agent is located in the .NET Core YAML file.

```
enable: true # Set to `true` for the agent to automatically upgrade to
newer versions.
checks: Startup # Set the frequency with which the agent checks for
updates. Valid values are `daily` for every 24 hours and on startup, or
`startup` for *only* when service starts up.
timeout_ms: 60000 # Set the time allocated to execute the downloaded agent
installer before cancelling.
nuget_repository_url: https://api.nuget.org/v3/index.json # Set the URL of
the Nuget repository to be used for the .NET Core Agent for IIS Installer
nuget_package_name: Contrast.CoreIIS.Installer # Set the name of the .NET
Core Agent for IIS Nuget package.
installer_upgrade_code: 82468c04-dfc0-4a4c-9eb9-c4b314c67fdc # Used
internally to retrieve the current installed agent version from Windows.
enable_major_version_upgrade: false # Set to `true` to automatically
upgrade major versions.
```



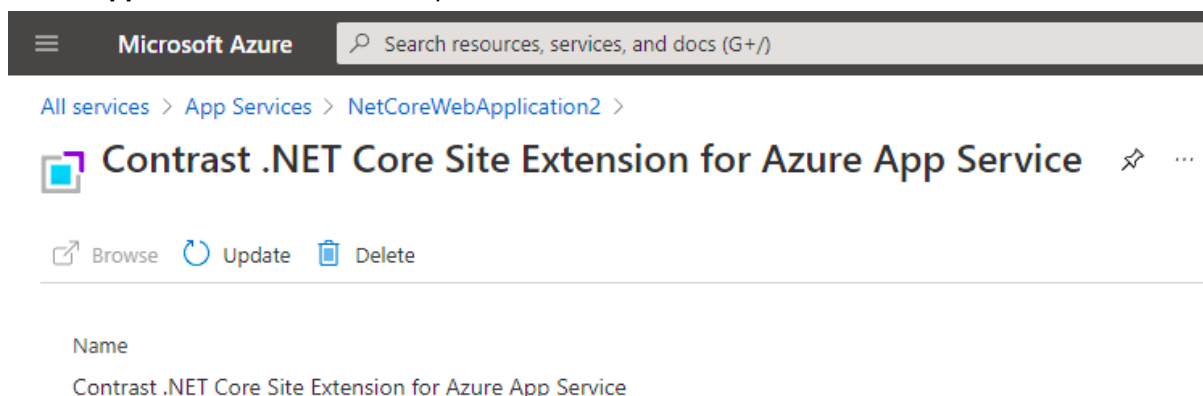
NOTE

The Agent Upgrade Service is only included with the agent installer. It is not included with the manual .NET Core Agent, agent NuGet packages, or Azure App Service site extensions.

Update the .NET Core agent

Contrast frequently releases new versions of agents, these steps show you how to easily update the NuGet package or the manual .NET Core agent and keep it updated. To update the agent by installation:

- **.NET Core agent agent for IIS installer:** use the [Agent upgrade service \(page 226\)](#).
- **Azure App Service:** Use the Azure portal.



- **Manual installation:** use the instructions below to set up your own automation.

Before you begin

- Confirmed your .NET Core application runs properly without the Contrast .NET Core agent.
- Previously installed the Contrast .NET Core agent.
- Defined a policy for how and when to update the agent, based on your change management policy and the environment where you deploy agents.
- An existing workflow to manage and keep application dependencies updated.

Steps

1. Download the Contrast .NET Core agent to the same installation location by using the Contrast repository:
 - Hosted: Contrast synchronizes .NET Core agent releases with public NuGet repositories.
 - On-premises: Contrast does not recommend using newer versions of Contrast agents than those available from your Contrast instance. Use the same version of the .NET Core agent version you would otherwise download directly from the Contrast web interface.
2. Get the following API information

```
CONTRAST_URL=<TeamServer URL e.g. https://app.contrastsecurity.com >  
ORG_ID=<YOUR TEAMSERVER ORGANIZATION ID>  
AUTH_TOKEN=<YOUR TEAMSERVER AUTHENTICATION TOKEN>  
API_KEY=<YOUR TEAMSERVER API KEY>
```

3. Use one of the following scripts to download Contrast .NET Core agent. Include the script in the application startup script, automated deployment pipeline, or add the script as a cron job to automatically update the agent.

- Bash script

```
CONTRAST_URL=https://app.contrastsecurity.com  
ORG_ID=xxxx  
AUTH_TOKEN=xxxx  
API_KEY=xxxx  
curl -X GET $CONTRAST_URL/Contrast/api/ng/$ORG_ID/agents/default/  
DOTNET_CORE /-o ./Contrast.NET.Core.zip -H 'Authorization:  
$AUTH_TOKEN' -H 'API-Key: $API_KEY' /-H 'Accept: application/json' -OJ
```

- Powershell

```
$ContrastUrl = "https://app.contrastsecurity.com/Contrast"  
$UserId = ""  
$ServiceKey = ""  
$ApiKey = ""  
$OrganizationId = ""  
$InstallPath = ".\dotnet-core"  
  
# Needed if the OS defaults to Tls1.1.  
[Net.ServicePointManager]::SecurityProtocol =  
[Net.SecurityProtocolType]::Tls12  
  
New-Item -ItemType Directory $InstallPath  
  
Invoke-WebRequest `   
    -Uri "$ContrastUrl/api/ng/$OrganizationId/agents/default/  
DOTNET_CORE" `   
    -Headers @{  
        "API-Key" = $ApiKey  
        "Authorization"  
= [Convert]::ToBase64String([System.Text.Encoding]::UTF8.GetBytes("$  
{UserId}:{ServiceKey}"))  
    } `   
    -OutFile "$InstallPath\Contrast.zip"  
  
Invoke-WebRequest -Uri `   
    "$ContrastUrl/api/ng/$OrganizationId/agents/external/default/  
DOTNET_CORE" `
```

```
-Headers @{
    "Accept"          = "text/yaml"
    "API-Key"         = $ApiKey
    "Authorization"   = [Convert]::ToBase64String([System.Text.Encoding]::UTF8.GetBytes("{UserId}:{ServiceKey}"))
} `
-OutFile "$InstallPath\contrast_security.yaml"

Expand-Archive "$InstallPath\Contrast.zip" -DestinationPath
$InstallPath
Remove-Item "$InstallPath\Contrast.zip"
```

4. Unzip the downloaded file and save the contents to the current Contrast agent location. If you do not know the location, you can look up the environment variables using the command for your system.

- Windows (64-bit)

```
echo %CORECLR_PROFILER_PATH_64%CORECLR_PROFILER_PATH_64
```

- Windows (32-bit)

```
CORECLR_PROFILER_PATH_32
```

- Linux (64-bit)

```
CORECLR_PROFILER_PATH_64
```

- Powershell

```
printenv CORECLR_PROFILER_PATH_64
```

Configure the .NET Core agent

The [standard configuration \(page 102\)](#) for all agents uses this [order of precedence \(page 106\)](#).

Depending on your situation, you can configure the .NET Core agent with:

- [Azure App Service \(page 291\)](#)
- [Environment variables \(page 291\)](#)
- [A YAML configuration file \(page 292\)](#)
- [Integrations \(page 1051\)](#)



TIP

Use the [Contrast agent configuration editor \(page 109\)](#) to create or upload a YAML configuration file, validate YAML and get setting recommendations.

.NET profiling and diagnostics variables

For .NET 8 and later, setting the `DOTNET_EnableDiagnostics` environment variable to 0 disables all diagnostics for the process, including profiling. This setting prevents the Contrast agent from working with your applications. Setting the `DOTNET_EnableDiagnostics_Profiler` environment variable to 0 disables profiling only, however, it also prevents the Contrast agent from working with your .NET 8 applications.

This behavior is different from the behavior in .NET 7 and earlier applications.

**NOTE**

COMPlus_EnableDiagnostics is an alias of DOTNET_EnableDiagnostics, so setting this variable to 0 has the same effect on the Contrast agent.

To turn off the diagnostics port only and keep profiling on, set these environment variables:

```
DOTNET_EnableDiagnostics=1
```

```
DOTNET_EnableDiagnostics_IPC=0
```

[.NET Environment variables](#) provides additional information about these environment variables.

Configure the .NET Core agent for Azure App Service

When using Azure App Service, you can configure the .NET Core agent with:

- **The Azure Portal:** Configure the .NET Core agent using [environment variables \(page 291\)](#). Add all settings to the **Application Settings** section of the **Configuration** blade using environment variable syntax.
- **Environment variables in a web.config file:** Place your overrides using the environment variable convention in the <environmentVariables> section of <aspNetCore> element.
- **A YAML configuration file (page 292):** Upload the file to your Azure web application by including it in your application deployment or using the Kudu console.
In the **Configuration\Application Settings** blade, add a new application setting called `CONTRAST_CONFIG_PATH` with a value that points to this file.
For example, to use the `contrast_security.yaml` file in the root of your application, add a new application setting with the key `CONTRAST_CONFIG_PATH` and value of `D:\Home\site\wwwroot\contrast_security.yaml` in **Configuration\Application Settings**. Application files in Azure App Service are deployed to `D:\home\site\wwwroot`.

See also

- [Install the .NET Core agent with Azure App Service \(page 278\)](#)

Configure .NET Core agent with environment variables

You can configure environment variables in several ways:

- Under IIS, [the web.config file can be used to configure application environment variables](#)
- Under Azure App services, the Azure platform provides a UI to configure the web site's environment variables.
- When developing, the `launchSettings.json` file can be used to configure the environment variables on launched applications.

**TIP**

You can convert any of the properties in the [.NET Core YAML template \(page 292\)](#) to environment variables.

- To change the agent's logging level (`agent.logger.level`) to "TRACE", add a setting with key `CONTRAST__AGENT__LOGGER__LEVEL` and value "TRACE".
- To change the agent's server name (`server.name`) to "MyServer", add a setting with key `CONTRAST__SERVER__NAME` and value "MyServer".

Here are some of the most common settings:

Environment variable	Purpose
CONTRAST__APPLICATION__NAME	Specify the application name reported to Contrast.
CONTRAST__APPLICATION__GROUP	Specify the access group for this application. (You must have already created access groups (page 1164) .)
CONTRAST__APPLICATION__SESSION_METADATA	Provide metadata which is used to create a new session ID in the Contrast web interface. Vulnerabilities discovered by the agent are associated with this new session.
CONTRAST__SERVER__NAME	Specify the server name reported to Contrast.
CONTRAST__SERVER__ENVIRONMENT	Specify in which environment the application is running (Development, QA and Production).

See the [.NET Core YAML template \(page 292\)](#) for a description of other available properties.

.NET Core YAML configuration template

Use this template to configure the .NET Core agent using a YAML configuration file. (Learn more about [YAML configuration \(page 107\)](#).)

Place your YAML file in the default location:

- **Windows:** C:/ProgramData/contrast/dotnet-core/contrast_security.yaml
- **Unix:** /etc/contrast/dotnet-core/contrast_security.yaml

```
#
=====
==
# Use the properties in this YAML file to configure a Contrast agent.
# Go to https://docs.contrastsecurity.com/en/order-of-precedence.html
# to determine the order of precedence for configuration values.
#
=====
==

# Use this setting if you want to temporarily disable a Contrast agent.
# Set to `true` to enable the agent; set to `false` to disable the agent.
# enable: true

#
=====
==
# api
# Use the properties in this section to connect the agent to the Contrast
# UI.
#
=====
==
api:

  # ***** REQUIRED *****
  # Set the URL for the Contrast UI.
  url: https://app.contrastsecurity.com/Contrast

  # ***** REQUIRED *****
  # Set the API key needed to communicate with the Contrast UI.
  api_key: NEEDS_TO_BE_SET
```



```
# ***** REQUIRED *****
# Set the service key needed to communicate with the Contrast
# UI. It is used to calculate the Authorization header.
service_key: NEEDS_TO_BE_SET

# ***** REQUIRED *****
# Set the user name used to communicate with the Contrast
# UI. It is used to calculate the Authorization header.
user_name: NEEDS_TO_BE_SET

# base64 encoded JSON object containing the `url`,
# `api_key`, `service_key`, and `user_name` config options,
# allowing them all to be set in a single variable.
# token: NEEDS_TO_BE_SET

# Set the version of the TLS protocol the agent uses to communicate with
the
# Contrast UI. The .NET agent default behavior is
(SecurityProtocolType.Tls
# | SecurityProtocolType.Tls11 | SecurityProtocolType.Tls12).
# tls_versions: tls|tls11|tls12

#
=====
# api.certificate
# Use the following properties for communication
# with the Contrast UI using certificates.
#
=====
# certificate:

# If set to `false`, the agent will ignore the
# certificate configuration in this section.
# enable: true

# Determine the location from which the agent loads a client
# certificate. Value options include `File` or `Store`.
# certificate_location: NEEDS_TO_BE_SET

# Set the absolute path to the client certificate's
# .CER file for communication with Contrast UI. The
# `certificate_location` property must be set to `File`.
# cer_file: NEEDS_TO_BE_SET

# Specify the name of certificate store to open. The
# `certificate_location` property must be set to `Store`.
# Value options include `AuthRoot`, `CertificateAuthority`,
# `My`, `Root`, `TrustedPeople`, or `TrustedPublisher`.
# store_name: NEEDS_TO_BE_SET

# Specify the location of the certificate store. The
# `certificate_location` property must be set to `Store`.
# Value options include `CurrentUser` or `LocalMachine`.
# store_location: NEEDS_TO_BE_SET
```

```
# Specify the type of value the agent uses to find the certificate
# in the collection of certificates from the certificate store.
# The `certificate_location` property must be set to `Store`.
# Value options include `FindByIssuerDistinguishedName`,
# `FindByIssuerName`, `FindBySerialNumber`,
# `FindBySubjectDistinguishedName`, `FindBySubjectKeyIdentifier`,
# `FindBySubjectName`, or `FindByThumbprint`.
# find_type: NEEDS_TO_BE_SET

# Specify the value the agent uses in combination with
# `find_type` to find a certification in the certificate store.
#
# Note - The agent will use the first certificate from
# the certificate store that matches this search criteria.
#
# find_value: NEEDS_TO_BE_SET

#
=====
# api.proxy
# Use the following properties for communication
# with the Contrast UI over a proxy.
#
=====
# proxy:

# Set value to `true` for the agent to communicate
# with the Contrast web interface over a proxy. Set
# value to `false` if you don't want to use the proxy.
# enable: NEEDS_TO_BE_SET

# Set the URL for your Proxy Server. The URL form is `scheme://
host:port`.
# url: NEEDS_TO_BE_SET

# Set the proxy user.
# user: NEEDS_TO_BE_SET

# Set the proxy password.
# pass: NEEDS_TO_BE_SET

# Set the proxy authentication type. Value
# options are `NTLM`, `Digest`, and `Basic`.
# auth_type: NEEDS_TO_BE_SET

#
=====
==
# agent
# Use the properties in this section to control the way and frequency
# with which the agent communicates to logs and the Contrast UI.
#
=====
==
```

```
# agent:

#
=====
# agent.route_coverage
# Use the following properties for the route-based coverage feature.
#
=====
# route_coverage:

# Set to `false` to stop the agent from sending
# route-based coverage data to the Contrast UI when the
#
# application returns an error code indicating
# the request was not processed as expected.
#
# report_on_error: false

#
=====
# agent.logger
# Define the following properties to set logging values.
# If the following properties are not defined, the
# agent uses the logging values from the Contrast UI.
#
=====
# logger:

# Set the the log output level. Valid options are
# `ERROR`, `WARN`, `INFO`, `DEBUG`, and `TRACE`.
# level: INFO

# Set to `true` to redirect all logs to
# `stdout` instead of the file system.
# stdout: false

# Set the roll size for log files in megabytes. The agent will
# attempt to prevent the log file from being larger than this size.
# roll_size: 100

# Set the number of backup files to keep. Set to `0` to disable.
# backups: 10

#
=====
# agent.security_logger
# Define the following properties to set security logging
# values associated with Protect. If not defined, the agent
# uses the security logging (CEF) values from the Contrast UI.
#
=====
# security_logger:

# Set the log level for security logging. Valid options
# are `ERROR`, `WARN`, `INFO`, `DEBUG`, and `TRACE`.
```

```
# level: ERROR

#
=====
# agent.security_logger.syslog
# Define the following properties to set Syslog values. If the
properties
# are not defined, the agent uses the Syslog values from the Contrast
UI.
#
=====
# syslog:

# Set to `true` to enable Syslog logging.
# enable: NEEDS_TO_BE_SET

# Set the IP address of the Syslog server
# to which the agent should send messages.
# ip: NEEDS_TO_BE_SET

# Set the port of the Syslog server to
# which the agent should send messages.
# port: NEEDS_TO_BE_SET

# Set the facility code of the messages the agent sends to Syslog.
# facility: 19

# Set the log level of Exploited attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_exploited: ALERT

# Set the log level of Blocked attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_blocked: NOTICE

# Set the log level of Blocked At Perimeter
# attacks. Value options are `ALERT`, `CRITICAL`,
# `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_blocked_perimeter: NOTICE

# Set the log level of Probed attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_probed: WARNING

# Set the log level of Suspicious attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_suspicious: WARNING

# Set the connection type used for Syslog messages.
# Value options are `UNENCRYPTED` and `ENCRYPTED`.
# connection_type: UNENCRYPTED

#
=====
# agent.dotnet
```

```
# The following properties apply to any .NET agent-wide configurations.
#
=====
# dotnet:

# Set a list of application pool names that the agent does not
# instrument or analyze. Names must be formatted as a comma-separated
# list. New after .NET Framework 19.1.3 and .NET Core 4.0.2.
# app_pool_denylist: NEEDS_TO_BE_SET

# Set a list of application pool names that the agent instruments or
# analyzes. If set, other application pools are ignored. Allowlist takes
# precedence over denylist. Names must be formatted as a comma-separated
# list. New after .NET Framework 19.1.3 and .NET Core 4.0.2.
# app_pool_allowlist: NEEDS_TO_BE_SET

# Set a list of application names that the agent does not
# analyze. (The applications are still instrumented).
# Names must be formatted as a comma-separated list.
# New after .NET Framework 19.1.3 and .NET Core 1.0.0.
# application_denylist: NEEDS_TO_BE_SET

# Set a list of application names that the agent analyzes.
# If set, other applications are not analyzed, but are
# still instrumented. Allowlist takes precedence over
# denylist. Names must be formatted as a comma-separated
# list. New after .NET Framework 19.1.3 and .NET Core 1.0.0.
# application_allowlist: NEEDS_TO_BE_SET

# Enable a profiler chaining feature to allow Contrast to
# work alongside other tools that use the CLR Profiling
# API. Defaults to `true`. New after .NET Framework 19.1.3
# (Installed Only) and .NET Core 1.9.3 (Installed Only).
# enable_chaining: true

# Indicate that the agent should monitor configuration files for
# changes. New after .NET Framework 50.0.15 and .NET Core 2.1.14.
# enable_file_watching: true

# Indicate that the agent should allow CLR optimizations
# of JIT-compiled methods. Defaults to `true`. New
# after .NET Framework 19.1.3 and .NET Core 1.0.0.
# enable_instrumentation_optimizations: true

# Indicate that the agent should allow the CLR to inline
# methods that are not instrumented by Contrast. Defaults to
# `true`. New after .NET Framework 19.1.3 and .NET Core 1.0.0.
# enable_jit_inlining: true

# Indicate that the agent should allow the CLR to perform
# transparency checks under full trust. Defaults to `false`.
# New after .NET Framework 19.1.3 and .NET Core 1.0.0.
# enable_transparency_checks: false

# Set to display ASCII art to std::out on agent startup. Defaults
```

```
# to `true`. New after .NET Framework 20.6.3 and .NET Core 1.0.0.
# enable_cat: true

# Sets the maximum amount of time a Protect regular expression
# is allowed to run before being cancelled. Set to -1 to never
# cancel regular expression execution. Defaults to `20_000`.
# New after .NET Framework 20.4.3 and .NET Core 1.5.0.
# protect_searchers_single_pattern_deadline_ms: 20_000

# Sets the maximum amount of time a 'Probe Analysis' Protect
# regular expression is allowed to run before being cancelled. Set
# to -1 to never cancel regular expression execution. Defaults to
# `5_000`. New after .NET Framework 20.7.3 and .NET Core 1.5.11.
# protect_searchers_probe_analysis_single_pattern_deadline_ms: 5_000

# Sets the maximum amount of time a Protect rule is
# allowed to run before being cancelled. Set to -1 to never
# cancel Protect rule execution. Defaults to `60_000`.
# New after .NET Framework 20.4.3 and .NET Core 1.5.0.
# protect_searchers_total_rule_deadline_ms: 60_000

# Sets the maximum amount of time a 'Probe Analysis' Protect
# rule is allowed to run before being cancelled. Set to -1 to
# never cancel Protect rule execution. Defaults to `10_000`.
# New after .NET Framework 20.7.3 and .NET Core 1.5.11.
# protect_searchers_probe_analysis_total_rule_deadline_ms: 10_000

# Sets the maximum duration of time agent log files should be kept
# since last write before being deleted by the agent. Defaults to
# `604_800_000`. New after .NET Framework 20.6.1 and .NET Core 1.5.5.
# log_cleanup_maximum_age_ms: 604_800_000

# Suppresses gathering process-level metrics (process level metrics are
# gathered by default), used to identify performance problems. Metric
# counters may further decrease the stability of already unstable
# systems and can be disabled (set to true) if issues occur. Defaults
# to `false`. New after .NET Framework 20.6.6 and .NET Core 1.5.10.
# suppress_metric_counters: false

# Enable file based application watching. Set to false if
# file watching is causing locking issues. Defaults to `true`.
# New after .NET Framework 20.7.3 and .NET Core 1.5.11.
# enable_file_based_app_watching: true

#
=====
==
# inventory
# Use the properties in this section to override the inventory features.
#
=====
==
# inventory:

# Set to `false` to disable inventory features in the agent.
```

```
# enable: true

# Apply a list of labels to libraries. Labels
# must be formatted as a comma-delimited list.
# Example - `label1, label2, label3`
#
# tags: NEEDS_TO_BE_SET

#
=====
==
# assess
# Use the properties in this section to control Assess.
#
=====
==
# assess:

# Include this property to determine if the Assess
# feature should be enabled. If this property is not
# present, the decision is delegated to the Contrast UI.
# enable: false

# Control the values captured by Assess vulnerability events. `Full`
# captures most values by calling ToString on objects, which can
# provide more info but causes increased memory usage. `Minimal`
# has better performance as it only captures String type objects
# as strings and uses type name for other object type values.
# event_detail: minimal

# Apply a list of labels to vulnerabilities and preflight
# messages. Labels must be formatted as a comma-delimited list.
# Example - `label1, label2, label3`
#
# tags: NEEDS_TO_BE_SET

# Value options are `ALL`, `SOME`, or `NONE`.
# stacktraces: ALL

#
=====
# assess.sampling
# Use the following properties to control sampling in the agent.
#
=====
# sampling:

# Set to `true` to enable sampling.
# enable: false

# This property indicates the number of requests
# to analyze in each window before sampling begins.
# baseline: 5

# This property indicates that every *nth*
```

```
# request after the baseline is analyzed.
# request_frequency: 10

# This property indicates the duration for which a sample set is valid.
# window_ms: 180_000

#
=====
# assess.rules
# Use the following properties to control simple rule configurations.
#
=====
# rules:

# Define a list of Assess rules to disable in the agent. To view a
# list of rule names, in Contrast go to user menu > Policy Management >
# Assess rules. The rules must be formatted as a comma-delimited list.
#
# Example - Set `reflected-xss,sql-injection` to disable
# the reflected-xss rule and the sql-injection rule.
#
# disabled_rules: NEEDS_TO_BE_SET

#
=====
==
# protect
# Use the properties in this section to override Protect features.
#
=====
==
# protect:

# Include this property to determine if the Protect
# feature should be enabled. If this property is not
# present, the decision is delegated to the Contrast UI.
# enable: false

#
=====
# protect.probe_analysis
# Use the settings in this section to
# control the behavior of probe analysis.
#
=====
# probe_analysis:

# Set to `false` to disable probe analysis.
# enable: true

#
=====
# protect.rules
# Use the following properties to set simple rule configurations.
#
```



```
=====
# rules:

# Define a list of Protect rules to disable in the agent. To view a
# list of rule names, in Contrast go to user menu > Policy Management >
# Protect rules. The rules must be formatted as a comma-delimited list.
# disabled_rules: NEEDS_TO_BE_SET

#
=====
# protect.rules.bot-blocker
# Use the following selection to configure if the
# agent blocks bots. Set to `true` to enable blocking.
#
=====
# bot-blocker:

# Set to `true` for the agent to block known bots.
# enable: false

#
=====
# protect.rules.sql-injection
# Use the following settings to configure the sql-injection rule.
#
=====
# sql-injection:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or off.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.sql-injection-semantic-chaining
# Use the following properties to configure how the
# sql injection semantic analysis chaining rule works.
#
=====
# sql-injection-semantic-chaining:

# Set the mode of the rule. Value options
# are `monitor`, `block` or `off`.
# mode: off

#
=====
# protect.rules.sql-injection-semantic-dangerous-functions
# Use the following properties to configure how the sql
# injection semantic analysis dangerous functions rule works.
#
```

```
=====
# sql-injection-semantic-dangerous-functions:

# Set the mode of the rule. Value options
# are `monitor`, `block` or `off`.
# mode: off

#
=====
# protect.rules.sql-injection-semantic-suspicious-unions
# Use the following properties to configure how the sql
# injection semantic analysis suspicious unions rule works.
#
=====
# sql-injection-semantic-suspicious-unions:

# Set the mode of the rule. Value options
# are `monitor`, `block` or `off`.
# mode: off

#
=====
# protect.rules.sql-injection-semantic-tautologies
# Use the following properties to configure how the sql
# injection semantic analysis tautologies rule works.
#
=====
# sql-injection-semantic-tautologies:

# Set the mode of the rule. Value options
# are `monitor`, `block` or `off`.
# mode: off

#
=====
# protect.rules.cmd-injection
# Use the following properties to configure
# how the command injection rule works.
#
=====
# cmd-injection:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

# Tell the agent to detect when commands come directly
# from input. The agent blocks if blocking is enabled.
# detect_phased_commands: true

#
```

```
=====
# protect.rules.cmd-injection-semantic-chained-commands
# Use the following properties to configure how the
# 'command injection - chained commands' rule works
#
=====
```

```
# cmd-injection-semantic-chained-commands:

# Set the mode of the rule. Value options
# are `monitor`, `block`, or `off`.
# mode: off

#
```

```
=====
# protect.rules.cmd-injection-semantic-dangerous-paths
# Use the following properties to configure how the
# 'command injection - dangerous paths' rule works
#
=====
```

```
# cmd-injection-semantic-dangerous-paths:

# Set the mode of the rule. Value options
# are `monitor`, `block`, or `off`.
# mode: off

#
```

```
=====
# protect.rules.cmd-injection-command-backdoors
# Use the following properties to configure how the
# 'command injection - command backdoors' rule works
#
=====
```

```
# cmd-injection-command-backdoors:

# Set the mode of the rule. Value options
# are `monitor`, `block`, or `off`.
# mode: off

#
```

```
=====
# protect.rules.path-traversal-semantic-file-security-bypass
# Use the following properties to configure how the
# 'path traversal - file security bypass' rule works
#
=====
```

```
# path-traversal-semantic-file-security-bypass:

# Set the mode of the rule. Value options
# are `monitor`, `block`, or `off`.
# mode: off

#
```

```
=====
# protect.rules.path-traversal
# Use the following properties to configure
```

```
# how the path traversal rule works.
#
=====
# path-traversal:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.method-tampering
# Use the following properties to configure
# how the method tampering rule works.
#
=====
# method-tampering:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.reflected-xss
# Use the following properties to configure how
# the reflected cross-site scripting rule works.
#
=====
# reflected-xss:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.unsafe-file-upload
# Use the following properties to configure
# how the unsafe file upload rule works.
#
=====
# unsafe-file-upload:
```

```
# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.xxe
# Use the following properties to configure
# how the XML external entity works.
#
=====
# xxe:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.untrusted-deserialization
# Use the following properties to configure
# how the untrusted deserialization rule works.
#
=====
# untrusted-deserialization:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
==
# application
# Use the properties in this section for
# the application(s) hosting this agent.
#
=====
==
# application:

# Override the reported application name.
```

```
#
# Note - On systems where multiple, distinct applications may be served
# by a single process, this configuration causes the agent to report
# all discovered applications as one application with the given name.
#
# Note - On Java systems where multiple, distinct applications may be
# served by a single process, this configuration causes the agent to
report
# all discovered applications as one application with the given name.
#
# name: NEEDS_TO_BE_SET

# Add the name of the application group with which this
# application should be associated in the Contrast UI.
# group: NEEDS_TO_BE_SET

# Add the application code this application should use in the Contrast UI.
# code: NEEDS_TO_BE_SET

# Override the reported application version.
# version: NEEDS_TO_BE_SET

# Apply labels to an application. Labels must
# be formatted as a comma-delimited list.
# Example - `label1,label2,label3`
#
# tags: NEEDS_TO_BE_SET

# Define a set of `key=value` pairs (which conforms to RFC 2253) for
# specifying user-defined metadata associated with the application. The
# set must be formatted as a comma-delimited list of `key=value` pairs.
# Example - `business-unit=accounting, office=Baltimore`
#
# metadata: NEEDS_TO_BE_SET

# Provide the ID of a session which already exists in the Contrast
# UI. Vulnerabilities discovered by the agent are associated with
# this session. If an invalid ID is supplied, the agent will be
# disabled. This option and `application.session_metadata` are
# mutually exclusive; if both are set, the agent will be disabled.
# session_id: NEEDS_TO_BE_SET

# Provide metadata which is used to create a new session ID in the
# Contrast UI. Vulnerabilities discovered by the agent are associated with
# this new session. This value should be formatted as `key=value` pairs
# (conforming to RFC 2253). Available key names for this configuration
# are branchName, buildNumber, commitHash, committer, gitTag, repository,
# testRun, and version. This option and `application.session_id` are
# mutually exclusive; if both are set the agent will be disabled.
# session_metadata: NEEDS_TO_BE_SET

#
=====
==
# server
```

```
# Use the settings in this section to set metadata for the server
# hosting this agent. Contrast recognizes common, supported server
# names, paths, types and environments. Doing this may require a new
# server or license, and it may affect functionality of some features.
#
```

```
=====
==
```

```
# server:
```

```
  # Override the reported server name.
  # name: localhost
```

```
  # Set the environment directly to override the default set
  # by the Contrast UI. This allows the user to configure the
  # environment dynamically at startup rather than manually
  # updating the Server in the Contrast UI themselves afterwards.
  #
```

```
  # Valid values include `QA`, `PRODUCTION` and `DEVELOPMENT`.
  # For example, `PRODUCTION` registers this Server as
  # running in a `PRODUCTION` environment, regardless of the
  # organization's default environment in the Contrast UI.
  #
```

```
  # environment: NEEDS_TO_BE_SET
```

```
  # Apply a list of labels to the server. Labels
  # must be formatted as a comma-delimited list.
  # Example - `label1,label2,label3`
  #
```

```
  # tags: NEEDS_TO_BE_SET
```

```
  # Override the reported server path. New after
  # .NET Framework v21.3.1 and .NET Core v1.8.0.
  # path: NEEDS_TO_BE_SET
```

```
#
```

```
=====
==
```

```
# Use the properties in this YAML file to configure a Contrast agent.
# Go to https://docs.contrastsecurity.com/en/order-of-precedence.html
# to determine the order of precedence for configuration values.
#
```

```
=====
==
```

```
# Use this setting if you want to temporarily disable a Contrast agent.
# Set to `true` to enable the agent; set to `false` to disable the agent.
# enable: true
```

```
#
```

```
=====
==
```

```
# api
```

```
# Use the properties in this section to connect the agent to the Contrast
UI.
```

```
#
=====
==
api:

# ***** REQUIRED *****
# Set the URL for the Contrast UI.
url: https://app.contrastsecurity.com/Contrast

# ***** REQUIRED *****
# Set the API key needed to communicate with the Contrast UI.
api_key: NEEDS_TO_BE_SET

# ***** REQUIRED *****
# Set the service key needed to communicate with the Contrast
# UI. It is used to calculate the Authorization header.
service_key: NEEDS_TO_BE_SET

# ***** REQUIRED *****
# Set the user name used to communicate with the Contrast
# UI. It is used to calculate the Authorization header.
user_name: NEEDS_TO_BE_SET

# base64 encoded JSON object containing the `url`,
# `api_key`, `service_key`, and `user_name` config options,
# allowing them all to be set in a single variable.
# token: NEEDS_TO_BE_SET

# Set the version of the TLS protocol the agent uses to communicate with
the
# Contrast UI. The .NET agent default behavior is
(SecurityProtocolType.Tls
# | SecurityProtocolType.Tls11 | SecurityProtocolType.Tls12).
# tls_versions: tls|tls11|tls12

#
=====
# api.certificate
# Use the following properties for communication
# with the Contrast UI using certificates.
#
=====
# certificate:

# If set to `false`, the agent will ignore the
# certificate configuration in this section.
# enable: true

# Determine the location from which the agent loads a client
# certificate. Value options include `File` or `Store`.
# certificate_location: NEEDS_TO_BE_SET

# Set the absolute path to the client certificate's
# .CER file for communication with Contrast UI. The
# `certificate_location` property must be set to `File`.
```



```
# cer_file: NEEDS_TO_BE_SET

# Specify the name of certificate store to open. The
# `certificate_location` property must be set to `Store`.
# Value options include `AuthRoot`, `CertificateAuthority`,
# `My`, `Root`, `TrustedPeople`, or `TrustedPublisher`.
# store_name: NEEDS_TO_BE_SET

# Specify the location of the certificate store. The
# `certificate_location` property must be set to `Store`.
# Value options include `CurrentUser` or `LocalMachine`.
# store_location: NEEDS_TO_BE_SET

# Specify the type of value the agent uses to find the certificate
# in the collection of certificates from the certificate store.
# The `certificate_location` property must be set to `Store`.
# Value options include `FindByIssuerDistinguishedName`,
# `FindByIssuerName`, `FindBySerialNumber`,
# `FindBySubjectDistinguishedName`, `FindBySubjectKeyIdentifier`,
# `FindBySubjectName`, or `FindByThumbprint`.
# find_type: NEEDS_TO_BE_SET

# Specify the value the agent uses in combination with
# `find_type` to find a certification in the certificate store.
#
# Note - The agent will use the first certificate from
# the certificate store that matches this search criteria.
#
# find_value: NEEDS_TO_BE_SET

#
=====
# api.proxy
# Use the following properties for communication
# with the Contrast UI over a proxy.
#
=====
# proxy:

# Set value to `true` for the agent to communicate
# with the Contrast web interface over a proxy. Set
# value to `false` if you don't want to use the proxy.
# enable: NEEDS_TO_BE_SET

# Set the URL for your Proxy Server. The URL form is `scheme://
host:port`.
# url: NEEDS_TO_BE_SET

# Set the proxy user.
# user: NEEDS_TO_BE_SET

# Set the proxy password.
# pass: NEEDS_TO_BE_SET

# Set the proxy authentication type. Value
```

```
# options are `NTLM`, `Digest`, and `Basic`.
# auth_type: NEEDS_TO_BE_SET

#
=====
==
# agent
# Use the properties in this section to control the way and frequency
# with which the agent communicates to logs and the Contrast UI.
#
=====
==
# agent:

#
=====
# agent.route_coverage
# Use the following properties for the route-based coverage feature.
#
=====
# route_coverage:

# Set to `false` to stop the agent from sending
# route-based coverage data to the Contrast UI when the
#
# application returns an error code indicating
# the request was not processed as expected.
#
# report_on_error: false

#
=====
# agent.logger
# Define the following properties to set logging values.
# If the following properties are not defined, the
# agent uses the logging values from the Contrast UI.
#
=====
# logger:

# Set the the log output level. Valid options are
# `ERROR`, `WARN`, `INFO`, `DEBUG`, and `TRACE`.
# level: INFO

# Set to `true` to redirect all logs to
# `stdout` instead of the file system.
# stdout: false

# Set the roll size for log files in megabytes. The agent will
# attempt to prevent the log file from being larger than this size.
# roll_size: 100

# Set the number of backup files to keep. Set to `0` to disable.
# backups: 10
```

```
#
=====
# agent.security_logger
# Define the following properties to set security logging
# values associated with Protect. If not defined, the agent
# uses the security logging (CEF) values from the Contrast UI.
#
=====
# security_logger:

# Set the log level for security logging. Valid options
# are `ERROR`, `WARN`, `INFO`, `DEBUG`, and `TRACE`.
# level: ERROR

#
=====
# agent.security_logger.syslog
# Define the following properties to set Syslog values. If the
properties
# are not defined, the agent uses the Syslog values from the Contrast
UI.
#
=====
# syslog:

# Set to `true` to enable Syslog logging.
# enable: NEEDS_TO_BE_SET

# Set the IP address of the Syslog server
# to which the agent should send messages.
# ip: NEEDS_TO_BE_SET

# Set the port of the Syslog server to
# which the agent should send messages.
# port: NEEDS_TO_BE_SET

# Set the facility code of the messages the agent sends to Syslog.
# facility: 19

# Set the log level of Exploited attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_exploited: ALERT

# Set the log level of Blocked attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_blocked: NOTICE

# Set the log level of Blocked At Perimeter
# attacks. Value options are `ALERT`, `CRITICAL`,
# `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_blocked_perimeter: NOTICE

# Set the log level of Probed attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_probed: WARNING
```

```
# Set the log level of Suspicious attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_suspicious: WARNING

# Set the connection type used for Syslog messages.
# Value options are `UNENCRYPTED` and `ENCRYPTED`.
# connection_type: UNENCRYPTED

#
=====
# agent.dotnet
# The following properties apply to any .NET agent-wide configurations.
#
=====
# dotnet:

# Set a list of application pool names that the agent does not
# instrument or analyze. Names must be formatted as a comma-separated
# list. New after .NET Framework 19.1.3 and .NET Core 4.0.2.
# app_pool_denylist: NEEDS_TO_BE_SET

# Set a list of application pool names that the agent instruments or
# analyzes. If set, other application pools are ignored. Allowlist takes
# precedence over denylist. Names must be formatted as a comma-separated
# list. New after .NET Framework 19.1.3 and .NET Core 4.0.2.
# app_pool_allowlist: NEEDS_TO_BE_SET

# Set a list of application names that the agent does not
# analyze. (The applications are still instrumented).
# Names must be formatted as a comma-separated list.
# New after .NET Framework 19.1.3 and .NET Core 1.0.0.
# application_denylist: NEEDS_TO_BE_SET

# Set a list of application names that the agent analyzes.
# If set, other applications are not analyzed, but are
# still instrumented. Allowlist takes precedence over
# denylist. Names must be formatted as a comma-separated
# list. New after .NET Framework 19.1.3 and .NET Core 1.0.0.
# application_allowlist: NEEDS_TO_BE_SET

# Enable a profiler chaining feature to allow Contrast to
# work alongside other tools that use the CLR Profiling
# API. Defaults to `true`. New after .NET Framework 19.1.3
# (Installed Only) and .NET Core 1.9.3 (Installed Only).
# enable_chaining: true

# Indicate that the agent should monitor configuration files for
# changes. New after .NET Framework 50.0.15 and .NET Core 2.1.14.
# enable_file_watching: true

# Indicate that the agent should allow CLR optimizations
# of JIT-compiled methods. Defaults to `true`. New
# after .NET Framework 19.1.3 and .NET Core 1.0.0.
# enable_instrumentation_optimizations: true
```

```
# Indicate that the agent should allow the CLR to inline
# methods that are not instrumented by Contrast. Defaults to
# `true`. New after .NET Framework 19.1.3 and .NET Core 1.0.0.
# enable_jit_inlining: true

# Indicate that the agent should allow the CLR to perform
# transparency checks under full trust. Defaults to `false`.
# New after .NET Framework 19.1.3 and .NET Core 1.0.0.
# enable_transparency_checks: false

# Set to display ASCII art to std::out on agent startup. Defaults
# to `true`. New after .NET Framework 20.6.3 and .NET Core 1.0.0.
# enable_cat: true

# Sets the maximum amount of time a Protect regular expression
# is allowed to run before being cancelled. Set to -1 to never
# cancel regular expression execution. Defaults to `20_000`.
# New after .NET Framework 20.4.3 and .NET Core 1.5.0.
# protect_searchers_single_pattern_deadline_ms: 20_000

# Sets the maximum amount of time a 'Probe Analysis' Protect
# regular expression is allowed to run before being cancelled. Set
# to -1 to never cancel regular expression execution. Defaults to
# `5_000`. New after .NET Framework 20.7.3 and .NET Core 1.5.11.
# protect_searchers_probe_analysis_single_pattern_deadline_ms: 5_000

# Sets the maximum amount of time a Protect rule is
# allowed to run before being cancelled. Set to -1 to never
# cancel Protect rule execution. Defaults to `60_000`.
# New after .NET Framework 20.4.3 and .NET Core 1.5.0.
# protect_searchers_total_rule_deadline_ms: 60_000

# Sets the maximum amount of time a 'Probe Analysis' Protect
# rule is allowed to run before being cancelled. Set to -1 to
# never cancel Protect rule execution. Defaults to `10_000`.
# New after .NET Framework 20.7.3 and .NET Core 1.5.11.
# protect_searchers_probe_analysis_total_rule_deadline_ms: 10_000

# Sets the maximum duration of time agent log files should be kept
# since last write before being deleted by the agent. Defaults to
# `604_800_000`. New after .NET Framework 20.6.1 and .NET Core 1.5.5.
# log_cleanup_maximum_age_ms: 604_800_000

# Suppresses gathering process-level metrics (process level metrics are
# gathered by default), used to identify performance problems. Metric
# counters may further decrease the stability of already unstable
# systems and can be disabled (set to true) if issues occur. Defaults
# to `false`. New after .NET Framework 20.6.6 and .NET Core 1.5.10.
# suppress_metric_counters: false

# Enable file based application watching. Set to false if
# file watching is causing locking issues. Defaults to `true`.
# New after .NET Framework 20.7.3 and .NET Core 1.5.11.
# enable_file_based_app_watching: true
```

```
#
=====
==
# inventory
# Use the properties in this section to override the inventory features.
#
=====
==
# inventory:

# Set to `false` to disable inventory features in the agent.
# enable: true

# Apply a list of labels to libraries. Labels
# must be formatted as a comma-delimited list.
# Example - `label1, label2, label3`
#
# tags: NEEDS_TO_BE_SET

#
=====
==
# assess
# Use the properties in this section to control Assess.
#
=====
==
# assess:

# Include this property to determine if the Assess
# feature should be enabled. If this property is not
# present, the decision is delegated to the Contrast UI.
# enable: false

# Control the values captured by Assess vulnerability events. `Full`
# captures most values by calling ToString on objects, which can
# provide more info but causes increased memory usage. `Minimal`
# has better performance as it only captures String type objects
# as strings and uses type name for other object type values.
# event_detail: minimal

# Apply a list of labels to vulnerabilities and preflight
# messages. Labels must be formatted as a comma-delimited list.
# Example - `label1, label2, label3`
#
# tags: NEEDS_TO_BE_SET

# Value options are `ALL`, `SOME`, or `NONE`.
# stacktraces: ALL

#
=====
# assess.sampling
# Use the following properties to control sampling in the agent.
```

```
#
=====
# sampling:

# Set to `true` to enable sampling.
# enable: false

# This property indicates the number of requests
# to analyze in each window before sampling begins.
# baseline: 5

# This property indicates that every *nth*
# request after the baseline is analyzed.
# request_frequency: 10

# This property indicates the duration for which a sample set is valid.
# window_ms: 180_000

#
=====
# assess.rules
# Use the following properties to control simple rule configurations.
#
=====
# rules:

# Define a list of Assess rules to disable in the agent. To view a
# list of rule names, in Contrast go to user menu > Policy Management >
# Assess rules. The rules must be formatted as a comma-delimited list.
#
# Example - Set `reflected-xss,sql-injection` to disable
# the reflected-xss rule and the sql-injection rule.
#
# disabled_rules: NEEDS_TO_BE_SET

#
=====
==
# protect
# Use the properties in this section to override Protect features.
#
=====
==
# protect:

# Include this property to determine if the Protect
# feature should be enabled. If this property is not
# present, the decision is delegated to the Contrast UI.
# enable: false

#
=====
# protect.probe_analysis
# Use the settings in this section to
# control the behavior of probe analysis.
```

```
#
=====
# probe_analysis:

# Set to `false` to disable probe analysis.
# enable: true

#
=====
# protect.rules
# Use the following properties to set simple rule configurations.
#
=====
# rules:

# Define a list of Protect rules to disable in the agent. To view a
# list of rule names, in Contrast go to user menu > Policy Management >
# Protect rules. The rules must be formatted as a comma-delimited list.
# disabled_rules: NEEDS_TO_BE_SET

#
=====
# protect.rules.bot-blocker
# Use the following selection to configure if the
# agent blocks bots. Set to `true` to enable blocking.
#
=====
# bot-blocker:

# Set to `true` for the agent to block known bots.
# enable: false

#
=====
# protect.rules.sql-injection
# Use the following settings to configure the sql-injection rule.
#
=====
# sql-injection:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or off.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.sql-injection-semantic-chaining
# Use the following properties to configure how the
# sql injection semantic analysis chaining rule works.
#
=====
```



```
# sql-injection-semantic-chaining:

# Set the mode of the rule. Value options
# are `monitor`, `block` or `off`.
# mode: off

#
=====
# protect.rules.sql-injection-semantic-dangerous-functions
# Use the following properties to configure how the sql
# injection semantic analysis dangerous functions rule works.
#
=====
# sql-injection-semantic-dangerous-functions:

# Set the mode of the rule. Value options
# are `monitor`, `block` or `off`.
# mode: off

#
=====
# protect.rules.sql-injection-semantic-suspicious-unions
# Use the following properties to configure how the sql
# injection semantic analysis suspicious unions rule works.
#
=====
# sql-injection-semantic-suspicious-unions:

# Set the mode of the rule. Value options
# are `monitor`, `block` or `off`.
# mode: off

#
=====
# protect.rules.sql-injection-semantic-tautologies
# Use the following properties to configure how the sql
# injection semantic analysis tautologies rule works.
#
=====
# sql-injection-semantic-tautologies:

# Set the mode of the rule. Value options
# are `monitor`, `block` or `off`.
# mode: off

#
=====
# protect.rules.cmd-injection
# Use the following properties to configure
# how the command injection rule works.
#
=====
# cmd-injection:

# Set the mode of the rule. Value options are
```

```
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

# Tell the agent to detect when commands come directly
# from input. The agent blocks if blocking is enabled.
# detect_phased_commands: true

#
=====
# protect.rules.cmd-injection-semantic-chained-commands
# Use the following properties to configure how the
# 'command injection - chained commands' rule works
#
=====
# cmd-injection-semantic-chained-commands:

# Set the mode of the rule. Value options
# are `monitor`, `block`, or `off`.
# mode: off

#
=====
# protect.rules.cmd-injection-semantic-dangerous-paths
# Use the following properties to configure how the
# 'command injection - dangerous paths' rule works
#
=====
# cmd-injection-semantic-dangerous-paths:

# Set the mode of the rule. Value options
# are `monitor`, `block`, or `off`.
# mode: off

#
=====
# protect.rules.cmd-injection-command-backdoors
# Use the following properties to configure how the
# 'command injection - command backdoors' rule works
#
=====
# cmd-injection-command-backdoors:

# Set the mode of the rule. Value options
# are `monitor`, `block`, or `off`.
# mode: off

#
=====
# protect.rules.path-traversal-semantic-file-security-bypass
# Use the following properties to configure how the
# 'path traversal - file security bypass' rule works
```

```
#
=====
# path-traversal-semantic-file-security-bypass:

# Set the mode of the rule. Value options
# are `monitor`, `block`, or `off`.
# mode: off

#
=====
# protect.rules.path-traversal
# Use the following properties to configure
# how the path traversal rule works.
#
=====
# path-traversal:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.method-tampering
# Use the following properties to configure
# how the method tampering rule works.
#
=====
# method-tampering:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.reflected-xss
# Use the following properties to configure how
# the reflected cross-site scripting rule works.
#
=====
# reflected-xss:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
```

```
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.unsafe-file-upload
# Use the following properties to configure
# how the unsafe file upload rule works.
#
=====
# unsafe-file-upload:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.xxe
# Use the following properties to configure
# how the XML external entity works.
#
=====
# xxe:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.untrusted-deserialization
# Use the following properties to configure
# how the untrusted deserialization rule works.
#
=====
# untrusted-deserialization:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off
```

```
#
=====
==
# application
# Use the properties in this section for
# the application(s) hosting this agent.
#
=====
==
# application:

# Override the reported application name.
#
# Note - On systems where multiple, distinct applications may be served
# by a single process, this configuration causes the agent to report
# all discovered applications as one application with the given name.
#
# Note - On Java systems where multiple, distinct applications may be
# served by a single process, this configuration causes the agent to
report
# all discovered applications as one application with the given name.
#
# name: NEEDS_TO_BE_SET

# Add the name of the application group with which this
# application should be associated in the Contrast UI.
# group: NEEDS_TO_BE_SET

# Add the application code this application should use in the Contrast UI.
# code: NEEDS_TO_BE_SET

# Override the reported application version.
# version: NEEDS_TO_BE_SET

# Apply labels to an application. Labels must
# be formatted as a comma-delimited list.
# Example - `label1,label2,label3`
#
# tags: NEEDS_TO_BE_SET

# Define a set of `key=value` pairs (which conforms to RFC 2253) for
# specifying user-defined metadata associated with the application. The
# set must be formatted as a comma-delimited list of `key=value` pairs.
# Example - `business-unit=accounting, office=Baltimore`
#
# metadata: NEEDS_TO_BE_SET

# Provide the ID of a session which already exists in the Contrast
# UI. Vulnerabilities discovered by the agent are associated with
# this session. If an invalid ID is supplied, the agent will be
# disabled. This option and `application.session_metadata` are
# mutually exclusive; if both are set, the agent will be disabled.
# session_id: NEEDS_TO_BE_SET

# Provide metadata which is used to create a new session ID in the
```

```
# Contrast UI. Vulnerabilities discovered by the agent are associated with
# this new session. This value should be formatted as `key=value` pairs
# (conforming to RFC 2253). Available key names for this configuration
# are branchName, buildNumber, commitHash, committer, gitTag, repository,
# testRun, and version. This option and `application.session_id` are
# mutually exclusive; if both are set the agent will be disabled.
# session_metadata: NEEDS_TO_BE_SET

#
=====
==
# server
# Use the settings in this section to set metadata for the server
# hosting this agent. Contrast recognizes common, supported server
# names, paths, types and environments. Doing this may require a new
# server or license, and it may affect functionality of some features.
#
=====
==
# server:

# Override the reported server name.
# name: localhost

# Set the environment directly to override the default set
# by the Contrast UI. This allows the user to configure the
# environment dynamically at startup rather than manually
# updating the Server in the Contrast UI themselves afterwards.
#
# Valid values include `QA`, `PRODUCTION` and `DEVELOPMENT`.
# For example, `PRODUCTION` registers this Server as
# running in a `PRODUCTION` environment, regardless of the
# organization's default environment in the Contrast UI.
#
# environment: NEEDS_TO_BE_SET

# Apply a list of labels to the server. Labels
# must be formatted as a comma-delimited list.
# Example - `label1,label2,label3`
#
# tags: NEEDS_TO_BE_SET

# Override the reported server path. New after
# .NET Framework v21.3.1 and .NET Core v1.8.0.
# path: NEEDS_TO_BE_SET
```

.NET Agent Explorer



IMPORTANT

Starting with .NET Core agent 4.0.0 and .NET Framework agent 51.0.0, Agent Explorer replaces the Contrast Tray application.

The .NET Agent Explorer is an application that displays high-level information about the health of the .NET Core and .NET Framework agents. Use this application to verify that the agent is working as expected, especially after you initially install the agent.

Installing an agent also installs this application. If you install both types of agents, only one instance of Agent Explorer is installed.

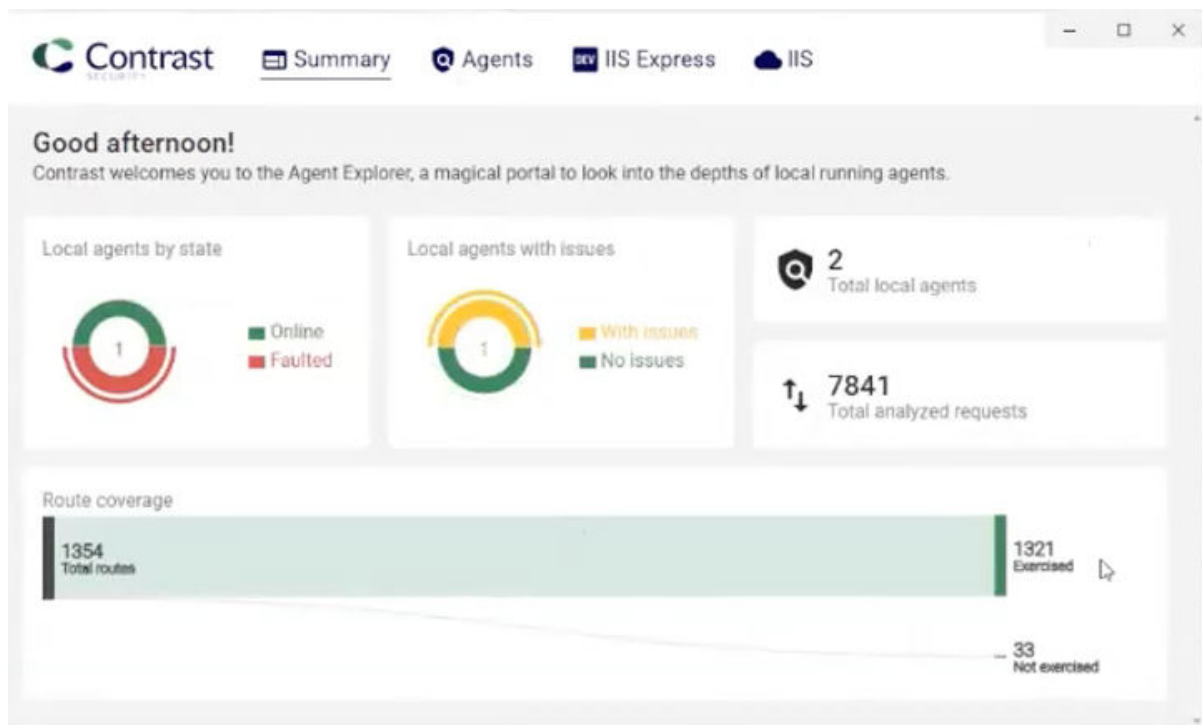
Agent Explorer access

After you install a .NET Core or .NET Framework agent, the Agent Explorer icon (🔍) displays in the tray. Right-click the icon to open the application.

Agent Explorer details

The Agent Explorer displays these details:

- **Summary**



This dashboard shows high level details about your agents, including their stage, whether any of them have issues, and route coverage.

- **Agents**

The screenshot shows the Contrast Security web interface. The top navigation bar includes 'Summary', 'Agents', 'IIS Express', and 'IIS'. The 'Overview' tab is selected, showing details for a .NET agent named 'DotnetCore5'. The details include:

- Application name: DotnetCore5
- Agent language: .NET Core
- Runtime version: .NET 5.0.17
- Total requests: 5152
- Running since: 19 minutes ago
- Application PID: 532188
- Agent version: 19.9.0.0
- Runtime bitness: x64
- Exercised routes: 1321/1354
- Agent modes: assess, protect, inventory
- YAML file path: [D:\Development\dotnet\dotnet-agent\tests\integration-tests\DotnetCore5\localAgent.yaml](#)
- Logs directory: [C:\ProgramData\contrast\dotnet-core\logs](#)

Below the details is a 'Configuration' section with a table of settings:

Name	Value	
enable	true	yaml-config
api.url	https://teamserver-...com	yaml-config

This tab provides details about the health of your .NET Core and .NET Framework agents. The Configuration section displays a message if Agent Explorer discovers a specific issue that is occurring.

You can access the agent's configuration (YAML) file directly from the link in the Overview section. Scroll down to see information about the agent's configuration, advanced information, and session metadata.

- **IIS Express**

This tab shows details for web applications running on IIS Express.

- **IIS**

This tab shows details for the web applications running on the IIS server.

Profiler chaining for the .NET Core agent

You can use profiler chaining to run the .NET Core agent alongside another .NET Core APM profiler.

The Contrast .NET Core agent is tested and proven to be compatible with the following profiling tools, given the combination of runtime, deployment type, and OS:

Profiling tool	Versions tested	.NET Core runtime	Third-party profiler deployment type	OS
AppDynamics	21.8.1	6.0	Installed, NuGet Package	Windows
Dynatrace One Agent	1.253.245	6.0	Installed	Windows, Linux
New Relic	8.23.107	6.0	NuGet Package	Windows, Linux
Riverbed SteelCentral Aternity APM	12.9.0	6.0	Installed	Windows
Datadog	2.35.0	6.0	Installed, NuGet Package	Windows, Linux



NOTE

The agent is likely compatible with other profiling tools if those tools follow the conventions of the CoreCLR Profiling API and do not make assumptions about the profiling environment.

Chaining is enabled by default and can be disabled by setting `agent.dotnet.enable_chaining` to `false`.

```
agent:
  dotnet:
    enable_chaining: false
```

Automatic (Windows and IIS)

When using the .NET Core installer for IIS, the installer configures chaining automatically for all hosted .NET Core applications.

1. Install the third-party agent first (recommended), then the Contrast .NET Core agent.
2. Restart the IIS workers (by default, this is done automatically by the agent installer). On restart, the Contrast .NET Core agent automatically detects the presence of other profiling tools registered with IIS and configures the environment to load both the Contrast .NET Core agent profiler and the third-party profiler.

Automatic (Linux)

Under Linux, automatic chaining may be configured by setting the `LD_PRELOAD` environment variable:

```
LD_PRELOAD=<path to the extracted Contrast files>/runtimes/linux-x64/native/ContrastChainLoader.so
```

For example, if the Contrast agent was extracted to `/contrast`, then the following would setup chaining automatically.

```
LD_PRELOAD=/contrast/runtimes/linux-x64/native/ContrastChainLoader.so
dotnet ./HelloWorld
```

Some APM profilers already set `LD_PRELOAD` (e.g. Dynatrace). In this case, ensure the Contrast module is loaded first - separate `LD_PRELOAD` are delimited by colons `:`. For example:

```
LD_PRELOAD=/contrast/runtimes/linux-x64/native/ContrastChainLoader.so:/
<path to Dynatrace>/liboneagentproc.so dotnet ./HelloWorld
```



NOTE

When running under Kubernetes, the [Contrast Agent Operator \(page 571\)](#) will automatically setup chaining and is the preferred route to add agents to existing Kubernetes workloads.

Manual

Chaining may need to be manually setup, for example, in the following environments:

- Hosted outside of IIS.
- Environments that use the third-party agent's Nuget package (rather than the installed agent).



NOTE

Chaining with the Dynatrace agent is only supported using the automatic options above.

1. Replace the CLR environment variables for the profiling tool with CONTRAST_CCC_CORECLR versions. Any of these names should be transformed:

Change this	To this
CORECLR_PROFILER	CONTRAST_CCC_CORECLR_PROFILER
CORECLR_PROFILER_PATH	CONTRAST_CCC_CORECLR_PROFILER_PATH
CORECLR_PROFILER_PATH_32	CONTRAST_CCC_CORECLR_PROFILER_PATH_32
CORECLR_PROFILER_PATH_64	CONTRAST_CCC_CORECLR_PROFILER_PATH_64

2. Then add the agent [manually \(page 273\)](#).

.NET Framework and .NET Core Telemetry

.NET Framework and .NET Core agents use telemetry to collect usage data. Telemetry is collected when an instrumented application first loads the agent's sensors and then periodically (every few hours) afterwards.

[Your privacy is important to us \(page 1275\)](#). The telemetry feature doesn't collect application data. The data is anonymized before being sent securely to Contrast. Then the aggregated data is stored encrypted and under restricted access control. Any collected data will be deleted after one year.

The telemetry feature collects the following data:

Agent versions	Data
.NET Framework later than 2020.8.3	Agent version
.NET Core later than 1.5.15	Operating system and version
	Whether the agent is running in a container
	Whether the agent is running in Azure App Service
	Hashed Media Access Control (MAC) address: a cryptographically (SHA256) anonymous and unique ID for a machine
	Kernel version
	Process running time
	Whether Assess is enabled
	Whether Protect is enabled
	.NET Framework later than 2020.8.3
	.NET Framework runtime version
	.NET Core later than 1.5.15
	.NET Core runtime version
.NET Framework later than 20.9.1	Hosted or on-premises Contrast instance
.NET Core later than 1.5.17	
.NET Framework later than 20.9.3	CLR Instrumentation Engine (CIE) usage
.NET Core later than 1.5.19	Application framework
	Chained profiler vendor
.NET Framework later than 20.10.1	Process hosting mode
.NET Core later than 1.5.20	CIE Raw Profiler Hook usage
.NET Framework later than 20.10.2	Names of configuration settings with non-default values
.NET Core later than 1.5.21	Names of disabled Assess rules
.NET Framework later than 20.12.2	Time elapsed for agent's profiler component to initialize
.NET Core later than 1.7.2	Time elapsed for agent's first request to the Contrast web interface
	Time elapsed for agent's profiler component to initialize
	Time elapsed between agent initialization and end of the first request

Agent versions	Data
.NET Framework later than 21.1.1	<p>Metrics on IIS-hosted applications, including:</p> <ul style="list-style-type: none"> • Total application count • Application count that will be analyzed (pass application allow list/deny list configuration) • Count of apps hosted on CLR4 application pools • Count of apps hosted on CLR2 application pools <p>Metrics on IIS applications pools</p> <ul style="list-style-type: none"> • Total count • Count with agent attached • Count of CLR4 • Count of CLR2 <p>Minimum number of applications in a single app pool</p> <p>Maximum number of applications in a single app pool</p> <p>Median number of applications across all app pools</p>
.NET Framework later than 21.1.2	Rule mode (i.e. Monitor vs. Block) for each Protect rule
.NET Core later than 1.7.5	
.NET Framework later than 21.4.2	Exceptions thrown and caught within agent sensor code, including log message, exception type, exception message, and stack trace frames for <code>System</code> and <code>Contrast</code> methods.
.NET Core later than 1.8.4	
.NET Framework later than 21.7.1	<ul style="list-style-type: none"> • Process Architecture (x86/x64)
.NET Core later than 1.9.7	<p>OS Architecture (x86/x64)</p> <p>In Azure App Service, the values of the following environment variables:</p> <ul style="list-style-type: none"> • <code>WEBSITE_PHYSICAL_MEMORY_MB</code> • <code>WEBSITE_PLATFORM_VERSION</code> • <code>WEBSITE_SKU</code>
.NET Framework later than 21.9.2	Description of location where YAML config file was loaded from (i.e., path specified by environment variable, default location, application directory).
.NET Core later than 2.0.1	

To opt-out of the telemetry feature, set the `CONTRAST_AGENT_TELEMETRY_OPTOUT` environment variable to `1` or `true`.

Telemetry data is securely sent to telemetry.dotnet.contrastsecurity.com. You can also opt out of telemetry by blocking communication at the network level.

Supported Azure functions

Versions

Runtime version	Language version	Supported	Not supported
1.x	.NET Framework 4.8	<ul style="list-style-type: none"> • Windows application: supported with Azure .NET Framework Site Extension 	<ul style="list-style-type: none"> • Docker Linux image • Linux application
3.x	.NET 5	<ul style="list-style-type: none"> • Windows Application: Supported locally and in Azure with the .NET Core Site Extension • Docker Linux Image: Supported locally and in Azure 	<ul style="list-style-type: none"> • Linux application
4.x	.NET 6	<ul style="list-style-type: none"> • Windows Application: Supported locally and in Azure with the .NET Core Site Extension • Docker Linux Image: Supported locally and in Azure 	<ul style="list-style-type: none"> • Linux application

**NOTE**

For all versions, running the Azure Functions application in isolated mode is not supported by the .NET agent.

Supported triggers

- HTTP
- Service Bus

Configuration

Azure Functions supports three deployment scenarios: Windows applications, Linux applications, and Docker Linux images. Of those three, only Windows applications and Docker Linux images are compatible with the agent. Linux application deployment is not supported by the .NET agent.

Windows Application

The Windows application deployment option is fully supported for Azure Functions versions 1, 3, and 4. Locally, the application can reference the Contrast .NET Core Agent NuGet package. On Azure, you must install the Contrast .NET Core Site Extension. When you deploy the Azure Function application using a tool (for example, Visual Studio or Core Tools), in a CI/CD pipeline or using the Azure Functions portal editor, the site extension will **not** automatically set the required application settings. You must specify the application settings manually.

To do this, set the following application settings (Settings > Configuration > Application settings) to enable the agent to attach:

```
CORECLR_ENABLE_PROFILING=1
CORECLR_PROFILER={8B2CE134-0948-48CA-A4B2-80DDAD9F5791}
CORECLR_PROFILER_PATH_32=C:\home\SiteExtensions\Contrast.NetCore.Azure.SiteExtension\ContrastNetCoreAppService-<version>\runtimes\win-x86\native\ContrastProfiler.dll
CORECLR_PROFILER_PATH_64=C:\home\SiteExtensions\Contrast.NetCore.Azure.SiteExtension\ContrastNetCoreAppService-<version>\runtimes\win-x64\native\ContrastProfiler.dll
```

Where <version> is the version of the agent in the form "0.0.0.0". For example, if the agent version is "2.1.8" the path would be:

```
C:\home\SiteExtensions\Contrast.NetCore.Azure.SiteExtension\ContrastNetCoreAppService-2.1.8.0\runtimes\win-x64\native\ContrastProfiler.dll.
```

Connection information for Contrast server can be supplied either using application settings or in a configuration file that is pointed to by an application setting.

Application settings

```
CONTRAST__API__USER_NAME=my_username
CONTRAST__API__SERVICE_KEY=my_service_key
CONTRAST__API__API_KEY=my_api_key
CONTRAST__API__URL=my_api_url
```

If Contrast server connection information is supplied via configuration file, the following application setting must be set:

```
CONTRAST_CONFIG_PATH=C:\home\site\wwwroot\contrast_security.yaml
```

Docker Linux image

The custom Linux image deployment option is supported for Azure Functions versions 3 and 4. A custom Linux image is required and it must contain the application and the Contrast .NET Core Agent NuGet package. Note that Linux applications, when not run from a custom Linux image, are not supported by the .NET agent.

The following application settings must be set in order for the agent to attach. These can either be in the image itself (as environment variables) or set as application settings (Settings > Configuration > Application settings).

```
CORECLR_ENABLE_PROFILING=1
CORECLR_PROFILER={8B2CE134-0948-48CA-A4B2-80DDAD9F5791}
CORECLR_PROFILER_PATH=/home/site/wwwroot/contrast/runtimes/linux-x64/native/ContrastProfiler.so
CORECLR_PROFILER_PATH_64=/home/site/wwwroot/contrast/runtimes/linux-x64/native/ContrastProfiler.so
CONTRAST_CORECLR_INSTALL_DIRECTORY=/home/site/wwwroot/bin/contrast/
```

Connection information for Contrast server can be supplied either using application settings/environment variables or in a configuration file that is pointed to by an application setting/environment variable.

Application setting/environment variables:

```
CONTRAST_API_USER_NAME=my_username
CONTRAST_API_SERVICE_KEY=my_service_key
CONTRAST_API_API_KEY=my_api_key
CONTRAST_API_URL=my_api_url
```

If the Contrast server connection information is supplied via configuration file, the following application setting/environment variable must be set: `CONTRAST_CONFIG_PATH=/home/site/wwwroot/contrast_security.yaml`

Node.js agent

The Contrast Node.js agent analyzes the behavior of Node.js web applications using established techniques, such as source-to-source compilation, to add Contrast sensors to an application before execution.



NOTE

The latest Node.js agent supports Assess (IAST), Protect (RASP), and SCA features.

The Contrast Node.js agent follows semantic versioning (major.minor.patch). The agent works best with these [supported technologies \(page 330\)](#) and these [system requirements \(page 333\)](#).

The Node.js agent rewrites the application code before startup using the Babel compiler. After starting up the agent patches the required functions for the [supported frameworks and modules \(page 330\)](#).

Once you [install the Node.js agent \(page 334\)](#), there are two primary source code transformations that it uses to monitor the behavior of your application:

- **AST transformation** is the process by which the agent creates an abstract syntax tree of a body of code, manipulates the tree and then creates new source code based on this syntax tree. The agent

goes through this process to handle scenarios in which function hooks won't work. For example, rewrites allow Contrast to add operator overloading to JavaScript so that it can properly track the flow of untrusted data.

- **Function hooks** take over the execution of a given function like `child_process.exec`, to collect data about its arguments and its return value, and send this data to the parts of the agent responsible for analysis. As a result, the agent enables certain functions to be self-reporting.

Contrast service



NOTE

Contrast service is only required for the Node.js agent version 4.X.X and earlier.

The [Contrast service \(page 549\)](#) is an executable which is packaged within the Node.js agent and runs in a separate process. With versions 4.X.X of the agent the Contrast service starts up automatically with the agent.

The service enables communication between the Node.js agent and Contrast. Like the agent, it can be [configured \(page 550\)](#) with [environment variables \(page 110\)](#) or a [YAML configuration file \(page 107\)](#). The Contrast service uses port 30555 as the default for HTTP communication between the agent and the service.

You can configure the port and communication protocol between the agent and service. Available protocols include HTTP, Linux socket (file descriptor), and gRPC. The service can be deployed one-for-one with the agent, or shared across a group of agents on a single server hosting multiple containers.

Supported technologies for Node.js

This page reflects the supported technologies and capabilities of the latest version available on [npmjs.com](#) unless otherwise specified in the notes.



NOTE

The Contrast Node.js agent may not function with versions of modules tagged as deprecated on [npmjs.com](#). Deprecated modules present a high security risk and may negatively impact the agent's function.

It also does not support applications that use bundlers like `webpack`, `parcel`, or `esbuild` to package or compress the server-side JavaScript code.

Technology	Supported versions	Notes
Language versions	<ul style="list-style-type: none"> JavaScript ECMAScript 5 JavaScript ECMAScript 6 ECMAScript modules (ESM) TypeScript 	<p>Contrast supports even numbered Node.js versions in "active LTS" or "maintenance" status.</p> <p>The Node.js LTS versions support these features for JavaScript ECMAScript5 and 6.</p> <p>TypeScript is only supported if the agent is configured to point to the compiled entry point for your application.</p>
System	<ul style="list-style-type: none"> Node.js LTS version 16, 18, 20, 22 Processor support - Apple M1/M2, Intel/AMD (AMD64) Operating System Support - Windows Server, Windows 10/11, MacOS, Linux (Debian, CentOS, etc) PM2 System requirements for the Node.js agent (page 333) 	The 5.X version of the Node.js agent is the only version that supports Node.js LTS 22.
NPM version	<ul style="list-style-type: none"> >= 8.5.5 	
Application frameworks	<ul style="list-style-type: none"> Express 4, 5 Fastify 3, 4, 5 Hapi 19, 20, 21 Koa 2.3 and later Restify 8, 9, 10, 11 	
Database drivers and object-relational mapping (ORM)	<ul style="list-style-type: none"> MarsDB. No longer maintained but required by the JuiceShop vulnerable app. Mongoose 6.X, 7.X, 8.X MongoDB 2.2.36, 3.3.0 and later, 4.X, 5.X. Compatible with database versions 4.X, 5.X, 6.X. MySQL2 2.0.0 and later. Compatible with MySQL database versions 5.6.51, 5.7.X and 8.0.X. MSSQL 6.4.0 and later Postgres driver 7.5.0 and later; 8.X Sequelize 5.X (this is deprecated by the maintainer); 6.X SQLite3 driver 4.X. Compatible with database versions 3.26.0 and later. This is mainly for JuiceShop and demo apps, SQLite is not a "production" database. 	<ul style="list-style-type: none"> MongoDB 2.2.36 is only supported because it is required by the NodeGoat vulnerable application. SQLite and MarsDB are not for use in production and are only supported to enable running and testing with the JuiceShop vulnerable application.
Validation packages/libraries	<ul style="list-style-type: none"> Class-validator 0.13.0 and later Joi 17 and later Validator 13 and later 	
Templating engines	<ul style="list-style-type: none"> Pug 3 EJS 3.X 	
Other packages/libraries	<ul style="list-style-type: none"> Express-session 1.15.6 , 1.16.0 and later 	
Other Node.js technologies	<ul style="list-style-type: none"> HTTP/2 GraphQL 	

Supported technologies for v4 Node.js (Legacy)



NOTE

The Contrast Node.js agent may not function with versions of modules tagged as deprecated on npmjs.com. Deprecated modules present a high security risk and may negatively impact the function of the agent.

It also does not support applications that use bundlers like webpack, parcel, or esbuild to package or compress the server-side JavaScript code.

Language version	
<ul style="list-style-type: none"> JavaScript ECMAScript 5 JavaScript ECMAScript 6 ECMAScript modules (ESM) TypeScript 	<p>Notes</p> <p>Contrast supports even numbered Node.js versions in "active LTS" or "maintenance" status.</p> <p>The Node.js LTS versions support these features for JavaScript ECMAScript5 and 6.</p> <p>The Contrast Node.js agent provides limited support for working with user apps that use ESM.</p> <p>TypeScript is only supported if the agent is configured to point to the compiled entry point for your application.</p>
Node.js Long-Term Support (LTS)	
<p>All versions in Active and Maintenance LTS status, currently:</p> <ul style="list-style-type: none"> 12* and 14* 16* (only for agent version 4.5.0 and later) 18 (only for agent version 4.25.0 and later) 	<p>Notes</p> <p>You should always use Node.js LTS versions that are active or in maintenance status.</p> <p>*Although the Contrast agent should function when running 12 LTS, 14 LTS, or 16 LTS they reached EOL at the end of April 2022, April 2023 and September 11, 2023, respectively. These EOL versions present serious security risks since they are no longer patched.</p> <p>The Node.js agent doesn't guarantee support for Node.js features classified as Experimental (Stability: 1). It also doesn't instrument the native <code>net</code> module. It only provides functionality for <code>HTTP(S)</code> application servers built using the supported application frameworks in this table.</p> <p><code>HTTP/2</code> is supported for the Node.js agent when using the Node.js core <code>HTTP/2</code> or <code>spdy</code> library.</p> <p>For customer applications using <code>HTTP/2</code> with Contrast Node.js agent, you must configure the agent to use <code>assess.enable_lazy_tracking: false</code>.</p> <p>Node.js version status is shown in Node.js Long-Term Support Release Schedule.</p>
<ul style="list-style-type: none"> 20 (only for agent version 4.33.0 and later) 	<p>The agent does not support the feature that allows applications to run with the <code>--experimental-permission</code> flag and with reduced permissions. Reduced permissions inactivate native modules, and if the agent is instrumented with reduced permissions, it will immediately crash.</p>
Node package manager (npm)	
<p>npm versions:</p> <ul style="list-style-type: none"> >=6.13.7 >=7.11.0 >= 8.5.5 	<p>The Node.js agent requires access to one of these npm versions to reliably report libraries to the Contrast UI. Versions 6 or 8 are preferred over version 7.</p>
Application frameworks	

<ul style="list-style-type: none"> • Express 4 • hapi 16*, 17*, 18*, 19*, 20 • Fastify 3 • Koa 2.3 and later • Kraken 2.2.0 and 2.3.0 • LoopBack 3*, 4 • Restify 8 • Sails 1.2.3 and later 	Notes *Deprecated by the maintainer, these libraries could present a security risk.
Database drivers and object-relational mapping (ORM)	
<ul style="list-style-type: none"> • DynamoDB (Assess only) AWS SDK for JavaScript: 2.X and 3.X • MongoDB 2.2.36*, 3.3.0 and later, 4.X. Compatible with database versions 3.6, 4.X, 5.X) • MySQL2 2.0.0 and later. Compatible with MySQL database versions 5.6.51, 5.7.X and 8.0.X. • Mongoose 5.X, 6.X • MSSQL 6.4.0 and later • Postgres driver 7.5.0 and later, 8.X • RethinkDB driver version 2.4.0 and later • Sequelize 5.X and 6.X • SQLite3 driver 4.X. Compatible with database versions 3.26.0 and later). 	Notes *Deprecated by the maintainer, the agent will still function but these libraries/versions present a security risk.
Validation modules	
<ul style="list-style-type: none"> • Joi 17 and later • Validator 13 and later • Class-validator 0.13.0 and later 	
Templating engines	
<ul style="list-style-type: none"> • Handlebars 4 • Pug 3 • EJS 2.6.2, 3.0.1 • Mustache 4.x and later 	
Other technologies	
<ul style="list-style-type: none"> • Express-session 1.15.6 , 1.16.0 and later 	
Test suite	
Node Test Benches	When changes are made to the Node.js agent, Contrast runs this battery of automated tests to ensure that it detects findings in supported technologies across all supported versions of Node. The Node Test Benches include tests that exercise the agent with all of our supported frameworks. Each framework within the monorepo is updated as Contrast adds more third-party library support to the agent.

System requirements for the Node.js agent



IMPORTANT

The Node.js agent now has limited support for running on Macs with the M1/M2 chip. One limitation is the Node.js agent does not yet support running `Alpine` based docker containers on the Apple M1/M2 (ARM64). Running `Slim` based Docker images is supported.

This page reflects the system requirements and capabilities of the latest version available on npmjs.com unless otherwise specified in the notes.

Before installing the Node.js agent confirm you can meet the following requirements:

- There is a deployed application with a `package.json` file to be analyzed, and the web application technology is supported by Contrast.
- The agent has network connectivity with the Contrast server.

Using the Node.js agent requires increasing the application's available CPU and memory due to the increased processing and analysis of inbound information. Using the Node.js agent will use more resources than your application on its own. CPU load will also increase but this is heavily influenced by the specific application architecture and existing CPU usage profile.

If you are using Assess, you should double the available memory in each container compared to what you would normally use without the Contrast Node.js agent.

Requirement	Recommended	Notes
CPU	<ul style="list-style-type: none"> • AMD 64, x86_64 and compatible • Apple M1/M2 ARM64 (limited support) 	
Operating system	<ul style="list-style-type: none"> • CentOS/RHEL 7.9, 8 and later • Ubuntu 14.04 LTS, 16.04 LTS, 18.04 LTS, 20.04 LTS, 22.04 LTS • Debian 9, 10, 11 • Windows • macOS 	
Process managers	<ul style="list-style-type: none"> • PM2: 4.5.0 and later and 5.1.0 and later 	<ul style="list-style-type: none"> • The Contrast Node.js agent supports running in both fork and cluster mode. • The start command used when running the Contrast Node.js agent must include the full path to the installed <code>@contrast/agent</code> module. For example: <pre>node --import /Users/michael/Dev/my-app/node_modules/@contrast/agent app.js</pre>
Containers	<ul style="list-style-type: none"> • Distroless containers 	<ul style="list-style-type: none"> • Distroless container images do not include a shell which allows executing operating system commands. The latest Contrast Node.js agent (5.19.0 or later) reports libraries and findings even when you install it in an application that runs in a distroless container image.

Install the Node.js agent

There are several ways to install the agent depending on your situation, but generally, this is the process:

1. Install the Node.js agent from npm by running the `npm install @contrast/agent` command in the application's root directory.
2. Set [authentication keys \(page 104\)](#).
3. Add a command to the `package.json` file to enable your application to run with the agent. Depending on the method of installation, one of the following commands can be added to the `package.json` file.

- For Node LTS 18.19.0 and later, use the `--import` command to run the agent.

```
node --import @contrast/agent app-main.js [app arguments]
```

- For Node LTS versions greater than or equal to 16.17.0 and less than 18.19.0, use `--loader` to start the application.

```
node --loader @contrast/agent app-main.js [app arguments]
```

- For Node LTS versions less than 16.17.0 use the legacy method for starting the application. This is also applicable to applications not using the ESM syntax.

```
node -r @contrast/agent app-main.js [app arguments]
```

4. Use your application as you normally would and verify that Contrast sees the application.

5. To avoid errors, follow these specific instructions depending on how your application is deployed:
 - [Install manually \(page 335\)](#)
 - [Install in a container \(page 335\)](#)
 - [Install with IBM Cloud \(page 344\)](#)

Install Node.js agent manually

To install or update the agent manually:

1. Install the latest version of the agent from [npm](#) by running this command from the application's root directory:

```
npm install @contrast/agent
```

Alternatively, if you use yarn, run this command to install the agent:

```
yarn add @contrast/agent
```

2. Set [authentication keys \(page 104\)](#) with [environment variables \(page 110\)](#). Make sure your Node.js application has access to the environment variables at runtime.

Alternatively, set the configuration with [YAML configuration \(page 107\)](#) using this [template \(page 353\)](#). Make sure the `contrast_security.yaml` is in the applications root directory.

3. Add this command to the scripts section of your application's `package.json` file:

```
"scripts": {  
  "contrast": "node --import @contrast/agent app-main.js [app  
arguments]",  
  "start": ...,  
  "test": ...  
}
```

4. Run your application with the agent:

```
npm run contrast
```



TIP

You can change this npm script to include, [other runtime configurations \(page 352\)](#) such as an alternate configuration file location.

5. Exercise your application by performing either manual or automated testing to ensure your application is functioning correctly with the agent installed.
6. Verify that your server is registered in Contrast and reports an instance of your application.

Install the Node.js agent using a container

Installing the Node.js agent in a container is essentially the same as the standard installation procedure, except that the installation occurs in a container. To follow best practices, you should use environment variables to configure the Contrast credentials.

Using environment variables is the most secure method for installing the Node.js agent in a container. Since containers often migrate through QA and production systems, it's a best practice to avoid hard-coding credentials in the container definition.

Before you begin

This topic provides general guidance for installing the Node.js agent in a containerized application, with Docker as an example.

You should have a basic understanding of how containers and related software work. You may need to adjust the instructions to meet your specific circumstances.

Install the agent

Install the Node.js agent using one of these options:

- **Add the agent to the application during development.** (recommended)

This way, the agent will be included with your application's `package.json`.

Use this command to populate the agent into your pipelines and container images.

```
npm install @contrast/agent
```

- **Add the agent to the Dockerfile.**

Add the agent at container build time if you prefer to maintain separate images for the application (with and without the Contrast agent).

Use this command to add the agent to your existing Dockerfile or into a new Dockerfile that uses your application's image as a base image.

```
npm install @contrast/agent
```

Configure the agent

Follow these instructions when configuring the Node.js agent for an application deployed into a container like Docker (otherwise, see more general information on [configuring the Node.js agent \(page 352\)](#)). Configuration for the Node.js agent follows this [order of precedence \(page 106\)](#).

With agent token

This variable is a base64 encoded JSON object containing the url, api_key, service_key, and user_name configuration settings, allowing you to set them in a single variable

1. Use the agent token:

```
CONTRAST__API__TOKEN
```

2. Use environment variables to set application-specific configuration. These can be ENV statements in the Dockerfile or they can be passed to the Docker run command with the `-e` option. See a [list of environment variables \(page 352\)](#) commonly used to set application-specific values. You can also refer to the [Contrast agent configuration editor \(page 109\)](#) to view a full list of variables. For example, you could use this command to build your container:

```
docker build -t my-app-image
```

And then, use these commands when you run the container:

```
docker run -p 3000:3000 --name my-app-instance \
-e "CONTRAST__API__TOKEN=your-api-token" \
My-app-image
```

The process to set environment variables when using a cloud provider typically involves using a secrets manager and then linking the values of those secrets to the environment variable.

With legacy settings

If your agent configuration refers to both the legacy settings and the agent token, (in environment variables or the YAML file), the legacy settings take precedence. Remove references to the legacy settings to use just the agent token value.

1. If you are using a Node.js agent version earlier than 5.15.0, the required variables are:

```
export CONTRAST__API__URL
export CONTRAST__API__API_KEY
```

```
export CONTRAST__API__SERVICE_KEY
export CONTRAST__API__USER_NAME
```

2. Use environment variables to set application-specific configuration. These can be ENV statements in the Dockerfile or they can be passed to the Docker run command with the `-e` option. See a [list of environment variables \(page 352\)](#) commonly used to set application-specific values. You can also refer to the [Contrast agent configuration editor \(page 109\)](#) to view a full list of variables. For example, you could use this command to build your container:

```
docker build -t my-app-image
```

And then, use these commands when you run the container:

```
docker run -p 3000:3000 --name my-app-instance \
-e "CONTRAST__API__URL=your-ts-url" \
-e "CONTRAST__API__API_KEY=your-api-key" \
-e "CONTRAST__API__SERVICE_KEY=your-service-key" \
-e "CONTRAST__API__USER_NAME=your-user-name" \
My-app-image
```

The process to set environment variables when using a cloud provider typically involves using a secrets manager and then linking the values of those secrets to the environment variable.

Run and verify

1. If you want to use the Node.js agent rewriter CLI, see the [agent rewriter CLI \(page 387\)](#) instructions.
2. You must preload the Contrast agent when you launch your application. Normally, you do this in the Dockerfile's CMD statement, but you can also use an npm script defined in the `package.json`. For example, if you normally start your application with:

```
CMD [ "node", "app" ]
```

Then you can use this command to run the application with Contrast:

```
CMD [ "node", "--import", "@contrast/agent", "app" ]
```

3. When the agent starts, it will try to connect to Contrast with [authentication keys \(page 104\)](#).



TIP

To protect the agent credentials, use the Docker secret and pass them as environment variables during deployment time. For example:

```
docker run
-e CONTRAST__API__
-e CONTRAST__API__API_KEY=<value>
-e CONTRAST__API__SERVICE_KEY=<value>
-e CONTRAST__API__USER_NAME=<value>
-e CONTRAST__SERVER__ENVIRONMENT=<value> image_with_contrast
```

4. Verify that Contrast is running by checking the activity in the container log. For example, log activity might look like this:

```
2025-01-02 10:54:46
{"level":30,"time":1735833286319,"tid":0,"pid":1,"hostname":"5d9ee61ac84f",
,"name":"contrast","msg":"Starting @contrast/agent v5.23.0"}
2025-01-02 10:54:46
{"level":30,"time":1735833286319,"tid":0,"pid":1,"hostname":"5d9ee61ac84f
```

```
" , "name": "contrast", "config": { "_errors": [], "enable": "true", "api": {
  "enable": "true", "url": "https://teamserver-darpa-agents.contsec.com/
Contrast", "api_key": "contrast-redacted-
api.api_key", "service_key": "contrast-redacted-
api.service_key", "user_name": "agent_b47d608e-1971-4560-961c-
a55bd4111b61@Alexstestorg", "token": "contrast-redacted-api.token", "proxy":
{ "enable": "false", "url": "undefined" } }, "agent":
{ "stack_trace_limit": "10", "stack_trace_filters": "agent-, @contrast, node-
agent", "diagnostics": { "enable": "true", "report_path": "/juice-
shop" }, "route_coverage": { "enable": "true" }, "reporters":
{ "file": "undefined" }, "heap_dump":
{ "enable": "false", "path": "contrast_heap_dumps", "delay_ms": "10000", "window
_ms": "10000", "count": "5" }, "polling":
{ "app_activity_ms": "30000", "app_settings_ms": "30000", "app_update_ms": "300
00", "server_settings_ms": "30000" }, "logger": { "path": "/juice-shop/
contrast.log", "level": "info", "append": "true", "stdout": "true" }, "security_l
ogger": { "path": "/juice-shop/
security.log", "level": "error", "stdout": "false", "syslog":
{ "enable": "false", "ip": "127.0.0.1", "port": "514", "facility": "19", "severity
_exploited": "alert", "severity_blocked": "notice", "severity_blocked_perimit
er": "notice", "severity_probed": "warning", "severity_suspicious": "warning" }
}, "node": { "app_root": "/juice-
shop", "cmd_ignore_list": "", "exclusive_entrypoint": "undefined", "rewrite":
{ "enable": "true", "cache": { "enable": "true", "path": "/juice-
shop/.contrast" } }, "source_maps": { "enable": "true" }, "library_usage":
{ "reporting": { "enable": "true", "interval_ms": "100" } }, "metrics":
{ "enable": "true", "warn_ms": "5000" }, "npm_path": "npm" }, "inventory":
{ "analyze_libraries": "true", "gather_metadata_via": "undefined" }, "assess":
{ "enable": "false", "probabilistic_sampling":
{ "enable": "false", "base_probability": "0.1", "route_monitor":
{ "enable": "true", "ttl_ms": "1800000" } }, "tags": "undefined", "stacktraces": "A
LL", "max_context_source_events": "150", "max_propagation_events": "500", "saf
e_positives":
{ "enable": "false" }, "trust_custom_validators": "false" }, "protect":
{ "enable": "false", "probe_analysis": { "enable": "true" }, "rules":
{ "disabled_rules": "", "cmd-injection": { "mode": "off" }, "cmd-injection-
command-backdoors": { "mode": "off" }, "cmd-injection-semantic-chained-
commands": { "mode": "off" }, "cmd-injection-semantic-dangerous-paths":
{ "mode": "off" }, "crypto-bad-mac": { "mode": "off" }, "crypto-bad-ciphers":
{ "mode": "off" }, "crypto-weak-randomness": { "mode": "off" }, "method-
tampering": { "mode": "off" }, "nosql-injection": { "mode": "off" }, "nosql-
injection-mongo": { "mode": "off" }, "path-traversal": { "mode": "off" }, "path-
traversal-semantic-file-security-bypass": { "mode": "off" }, "reflected-xss":
{ "mode": "off" }, "sql-injection": { "mode": "off" }, "ssjs-injection":
{ "mode": "off" }, "ssrf": { "mode": "off" }, "unsafe-code-execution":
{ "mode": "off" }, "unsafe-file-upload": { "mode": "off" }, "untrusted-
deserialization": { "mode": "off" }, "xxe": { "mode": "off" }, "unvalidated-
redirect": { "mode": "off" } } }, "application": { "name": "mw-juice-
slim", "path": "/", "group": "undefined", "code": "undefined", "version": "undefi
ned", "tags": "undefined", "metadata": "undefined", "session_id": "undefined", "
session_metadata": "undefined" }, "server":
{ "name": "5d9ee61ac84f", "type": "Node.js
v20.15.0", "environment": "undefined", "tags": "undefined", "version": "undefin
ed", "discover_cloud_resource": "true" } }, "msg": "Agent configuration" }
2025-01-02 10:54:46
```

```
{ "level": 30, "time": 1735833286328, "tid": 0, "pid": 1, "hostname": "5d9ee61ac84f", "name": "contrast", "strategy": 1, "msg": "updating assess sampler" }
2025-01-02 10:54:48
{ "level": 30, "time": 1735833287906, "tid": 0, "pid": 1, "hostname": "5d9ee61ac84f", "name": "contrast", "msg": "Received new log level: debug from server-features" }
2025-01-02 10:54:48
{ "level": 30, "time": 1735833287907, "tid": 0, "pid": 1, "hostname": "5d9ee61ac84f", "name": "contrast", "msg": "assess sampling disabled" }
2025-01-02 10:54:48
{ "level": 30, "time": 1735833287907, "tid": 0, "pid": 1, "hostname": "5d9ee61ac84f", "name": "contrast", "enabledRules": [ "bot-blocker", "cmd-injection", "cmd-injection-command-backdoors", "cmd-injection-semantic-chained-commands", "cmd-injection-semantic-dangerous-paths", "crypto-bad-mac", "crypto-bad-ciphers", "crypto-weak-randomness", "ip-denylist", "method-tampering", "nosql-injection", "nosql-injection-mongo", "path-traversal", "path-traversal-semantic-file-security-bypass", "reflected-xss", "sql-injection", "ssjs-injection", "ssrf", "unsafe-code-execution", "unsafe-file-upload", "untrusted-deserialization", "virtual-patch", "xxe", "unvalidated-redirect", "autocomplete-missing", "cache-controls-missing", "clickjacking-control-missing", "parameter-pollution", "csp-header-missing", "csp-header-insecure", "hsts-header-missing", "x-powered-by-header", "xcontenttype-header-missing", "xxssprotection-header-disabled", "httponly", "secure-flag-missing" ], "msg": "Assess policy enabled rules updated" }
2025-01-02 10:54:48
{ "level": 30, "time": 1735833287907, "tid": 0, "pid": 1, "hostname": "5d9ee61ac84f", "name": "contrast", "policy": { "exclusions": { "url": [], "querystring": [], "header": [], "body": [], "cookie": [], "parameter": [] }, "bot-blocker": "block_at_perimeter", "cmd-injection": "monitor", "cmd-injection-command-backdoors": "monitor", "cmd-injection-semantic-chained-commands": "monitor", "cmd-injection-semantic-dangerous-paths": "monitor", "crypto-bad-mac": "off", "crypto-bad-ciphers": "off", "crypto-weak-randomness": "off", "ip-denylist": "off", "method-tampering": "monitor", "nosql-injection": "monitor", "nosql-injection-mongo": "monitor", "path-traversal": "monitor", "path-traversal-semantic-file-security-bypass": "monitor", "reflected-xss": "monitor", "sql-injection": "monitor", "ssjs-injection": "monitor", "ssrf": "off", "unsafe-code-execution": "off", "unsafe-file-upload": "off", "untrusted-deserialization": "monitor", "virtual-patch": "off", "xxe": "monitor", "unvalidated-redirect": "off", "rulesMask": 510 }, "msg": "Protect policy updated" }
2025-01-02 10:54:53
{ "level": 30, "time": 1735833293571, "tid": 2, "pid": 1, "hostname": "5d9ee61ac84f", "name": "contrast", "msg": "Received new log level: debug from server-features" }
2025-01-02 10:54:53
{ "level": 30, "time": 1735833293437, "tid": 0, "pid": 1, "hostname": "5d9ee61ac84f", "name": "contrast", "msg": "The Contrast Node Agent collects usage data in order to help us improve compatibility and security coverage.\nThe data is anonymous and does not contain application data. It is collected by Contrast and is never shared.\nYou can opt-out of telemetry by setting the CONTRAST_AGENT_TELEMETRY_OPTOUT environment variable to '1' or 'true'.\n" }
```

```
2025-01-02 10:54:53
{"level":20,"time":1735833293504,"tid":0,"pid":1,"hostname":"5d9ee61ac84f",
,"name":"contrast","msg":"deploying function toString patch"}
2025-01-02 10:55:00 info: Detected Node.js version v20.15.0 (OK)
2025-01-02 10:55:00 info: Detected OS linux (OK)
2025-01-02 10:55:00 info: Detected CPU arm64 (OK)
....
```

**NOTE**

You can also [install the agent when creating the Docker image \(page 340\)](#) [\(page 342\)](#) or [install a distroless Node.js container \(page 343\)](#) in containers

See also

Contrast Support Portal [Node.js agent with Kubernetes](#)

Contrast Support Portal [AWS Fargate and Contrast agents](#)

Install the agent when creating the Docker image**NOTE**

This procedure applies to version 5 and later of the Node.js agent.

Another option for installing the Contrast agent for a Node.js app is to run the `npm install` command as part of the Docker image creation instead of changing the `package.json` file in the source code repository.

This may be more desirable if you only want to modify the Docker file to be able to run a security test with the agent.

This procedure uses the [OWASP JuiceShop](#) vulnerable web app as an example.

Example:

```
FROM node:20-buster as installer
COPY . /juice-shop
WORKDIR /juice-shop
RUN npm i -g typescript ts-node
RUN npm install --omit=dev --unsafe-perm

# Install the latest Contrast agent and the cli rewriter
RUN npm install @contrast/agent@latest
RUN npm install --save-dev @contrast/cli

# Environment variables for the Contrast agent
ENV CONTRAST__AGENT__LOGGER__STDOUT=true
ENV CONTRAST__AGENT__LOGGER__PATH=/dev/null
```



```
# Take note that the following is optional and the var name has changed
from what was used by the v4 agent
ENV CONTRAST__AGENT__NODE__REWRITE__CACHE__PATH="/juice-shop/rewrite_cache"

# Assumes this project is rewriting for Assess only
ENV CONTRAST__ASSESS__ENABLE=true

# If no environment setting is specified the rewriter rewrites Protect
only. See the documentation to other settings.
RUN npx -p @contrast/cli rewrite build/app.js

RUN npm dedupe --omit=dev
RUN rm -rf frontend/node_modules
RUN rm -rf frontend/.angular
RUN rm -rf frontend/src/assets
RUN mkdir logs
RUN chown -R 65532 logs
RUN chgrp -R 0 ftp/ frontend/dist/ logs/ data/ i18n/
RUN chmod -R g=u ftp/ frontend/dist/ logs/ data/ i18n/
RUN rm data/chatbot/botDefaultTrainingData.json || true
RUN rm ftp/legal.md || true
RUN rm i18n/*.json || true

ARG CYCLONEDX_NPM_VERSION=latest
RUN npm install -g @cyclonedx/cyclonedx-npm@$CYCLONEDX_NPM_VERSION
RUN npm run sbom

# workaround for libxmljs startup error
FROM node:20-buster as libxmljs-builder
WORKDIR /juice-shop
RUN apt-get update && apt-get install -y build-essential python3
COPY --from=installer /juice-shop/node_modules ./node_modules
RUN rm -rf node_modules/libxmljs/build && \
  cd node_modules/libxmljs && \
  npm run build

FROM node:20-buster-slim
ARG BUILD_DATE
ARG VCS_REF

WORKDIR /juice-shop
COPY --from=installer /juice-shop .
COPY --from=libxmljs-builder /juice-shop/node_modules/libxmljs ./
node_modules/libxmljs
EXPOSE 3000

# Contrast logs will be written to the container
# This sets the rewrite cache path to match what was specified in
previously created image. Also take note that the following is optional
(either do not set on both places or set in both places) and the var name
has changed from what is used by the v4.x agent
ENV CONTRAST__AGENT__NODE__REWRITE__CACHE__PATH="/juice-shop/rewrite_cache"

# The following explicitly turns on Assess mode
ENV CONTRAST__ASSESS__ENABLE=true
```

```
# The start command has been modified to load and run the agent
CMD ["node", "--import", "@contrast/agent", "build/app.js"]
```

Install the agent when creating the Docker image (Legacy)



NOTE

This procedure applies to version 4 and earlier of the Node.js agent.

Another option for installing the Contrast agent for a Node.js app is to run the `npm install` command as part of the Docker image creation instead of changing the `package.json` file in the source code repository.

This may be more desirable if you only want to modify the Docker file to be able to run a security test with the agent.

Example:

```
FROM node:18 as installer
COPY . /juice-shop
WORKDIR /juice-shop
RUN npm i -g typescript ts-node
RUN npm install --omit=dev --unsafe-perm
RUN npm install @contrast/agent@4.x
RUN npm dedupe

# Needed to explicitly set Assess mode
ENV CONTRAST__APPLICATION__NAME=juice-assess-docker-slim
ENV CONTRAST__ASSESS__ENABLE=true
ENV CONTRAST__AGENT__LOGGER__STDOUT=true
ENV CONTRAST__AGENT__LOGGER__PATH=/dev/null
ENV DEBUG="contrast:*"

ENV CONTRAST__AGENT__NODE__REWRITE_CACHE__PATH="/juice-shop/rewrite_cache"

RUN npx contrast-transpile build/app.js

RUN rm -rf frontend/node_modules
RUN rm -rf frontend/.angular
RUN rm -rf frontend/src/assets
RUN mkdir logs
RUN chgrp -R 0 ftp/ frontend/dist/ logs/ data/ i18n/
RUN chmod -R g=u ftp/ frontend/dist/ logs/ data/ i18n/
#RUN rm data/chatbot/botDefaultTrainingData.json || true
#RUN rm ftp/legal.md || true
#RUN rm i18n/*.json || true

FROM node:18-slim
ARG BUILD_DATE
ARG VCS_REF

WORKDIR /juice-shop
```

```
COPY --from=installer /juice-shop .

EXPOSE 3000
# The following environment variables were added
ENV CONTRAST__APPLICATION__NAME=juice-assess-docker-slim
ENV CONTRAST__AGENT__SERVICE__GRPC=true
ENV CONTRAST__AGENT__LOGGER__STDOUT=true
ENV CONTRAST__AGENT__LOGGER__PATH=/dev/null
ENV DEBUG="contrast:*"

ENV CONTRAST__AGENT__NODE__REWRITE_CACHE__PATH="/juice-shop/rewrite_cache"

# This explicitly turns on Assess mode
ENV CONTRAST__ASSESS__ENABLE=true
ENV CONTRAST__ASSESS__ENABLE__LAZY_TRACKING=false
ENV CONTRAST__AGENT__NODE__APP_ROOT=/juice-shop

CMD ["node", "-r", "@contrast/agent", "build/app.js"]
```

Distroless containers

If using a *distroless* Node.js container then there is no **npm** or **shell** installed in the container image. You must use the `NODE_OPTIONS` environment variable to run the agent as a required module.

However, be careful when using `NODE_OPTIONS` since this will run the agent with all **node** or **npm** commands and may result in unintended execution resulting in longer start-up times.



NOTE

The latest Contrast Node.js agent reports libraries and findings even when installed in an application that runs in a distroless container image.

Example:

```
FROM node:18 as installer
COPY . /juice-shop
WORKDIR /juice-shop
RUN npm i -g typescript ts-node
RUN npm install --omit=dev --unsafe-perm
# install the latest agent
RUN npm install @contrast/agent

RUN npm dedupe

RUN rm -rf frontend/node_modules
RUN rm -rf frontend/.angular
RUN rm -rf frontend/src/assets
RUN mkdir logs
RUN chown -R 65532 logs
RUN chgrp -R 0 ftp/ frontend/dist/ logs/ data/ i18n/
RUN chmod -R g=u ftp/ frontend/dist/ logs/ data/ i18n/
RUN rm data/chatbot/botDefaultTrainingData.json || true
RUN rm ftp/legal.md || true
```

```
RUN rm il8n/*.json || true

FROM gcr.io/distroless/nodejs:18
ARG BUILD_DATE
ARG VCS_REF
LABEL maintainer="Bjoern Kimminich <bjoern.kimminich@owasp.org>" \
    org.opencontainers.image.title="OWASP Juice Shop" \
    org.opencontainers.image.description="Probably the most modern and
sophisticated insecure web application" \
    org.opencontainers.image.authors="Bjoern Kimminich
<bjoern.kimminich@owasp.org>" \
    org.opencontainers.image.vendor="Open Web Application Security Project" \
    org.opencontainers.image.documentation="https://help.owasp-juice.shop" \
    org.opencontainers.image.licenses="MIT" \
    org.opencontainers.image.version="14.5.1" \
    org.opencontainers.image.url="https://owasp-juice.shop" \
    org.opencontainers.image.source="https://github.com/juice-shop/juice-
shop" \
    org.opencontainers.image.revision=$VCS_REF \
    org.opencontainers.image.created=$BUILD_DATE
WORKDIR /juice-shop
COPY --from=installer --chown=65532:0 /juice-shop .
USER 65532
EXPOSE 3000

ENV NODE_OPTIONS "--import @contrast/agent"
CMD ["/juice-shop/build/app.js"]
```

Install Node.js with IBM Cloud

1. [Install the latest LTS \(Long Term Support\) version of Node.js.](#)
2. To install from [npm](#), run this command from the app root directory:

```
npm install @contrast/agent
```

Alternatively, if you use yarn, run this command to install the agent:

```
yarn add @contrast/agent
```

3. [Configure the Node.js \(page 352\)](#) using a YAML configuration file to set the [authentication keys \(page 104\)](#) and any application-specific configuration.

You can use this sample `contrast_security.yaml` file, but replace `<URL>`, `<UserName>`, `<APIKey>` and `<ServiceKey>` with your values, and set `<ServerName>` to the name of the IBM cloud server to which this application will report. (This way you will be able to identify the server when you view it in Contrast.)

```
contrast:
  url: <URL>
  user_name: <UserName>
  api_key: <APIKey>
  service_key: <ServiceKey>
server:
  name: <ServerName>
```

4. Copy your Contrast YAML file to your application's root directory or use environmental variables to set the required Contrast API credentials and configuration settings.
5. Add this command to the "scripts": section of your application's `package.json` file:

```
"ibmcloud-with-contrast": "CONTRAST_CONFIG_PATH=[the full path location of your YAML file] node --import @contrast/agent index.js",
```

6. Since IBM Cloud runs the start script by default, you must change the start command to point to the `ibmcloud-with-contrast` line given in the previous step. Run the agent using:

```
"start": "npm run ibmcloud-with-contrast"
```

Now the scripts section of the package.json should look like the following:

```
"scripts": {  
  "bluemix-with-contrast": "CONTRAST_CONFIG_PATH=[the full path location of your YAML file] node --import @contrast/agent index.js",  
  "start": "npm run bluemix-with-contrast"  
},
```

7. Push the application to IBM Cloud using:

```
cf push <application-name> -t 180
```

8. Run the agent with:

```
npm start
```

9. Exercise your application by performing either manual or automated testing to ensure your application is functioning correctly with the agent installed.
10. Verify that your server is registered in Contrast and reports an instance of your application.

Install Node.js agent with VMware Tanzu

You can access a variety of VMware Tanzu (formerly Pivotal Cloud Foundry) integrations for your applications using the default Node.js buildpack.

To use the buildpack on its own as a low-level integration, you can create a user-provided service and bind it to your application. With the service broker, you can define multiple service plans and generate service instances you can bind to your applications.

Use the [Contrast service broker tile \(page 346\)](#) to automate the BOSH deployment and configuration of the [Contrast service broker \(page 348\)](#).



IMPORTANT

The Contrast VMware Tanzu integration does not download the Node.js agent and modify your application startup. You must still download and [install the Node.js agent manually \(page 335\)](#).

You can configure the agent through the Contrast service broker tile provided with the integration or you can use automatic configuration through user-provided services.

Buildpacks

To install the Node.js agent in a VMware Tanzu environment, your application must use one of these buildpacks:

- **For tile support:** [NodeJS Buildpack](#) version 1.6.52 and later
- **For user-provided service support:** [NodeJS Buildpack](#) version 1.6.56 and later

If you are using a buildpack that does not include Contrast Security framework support, you can add it. To do this, you must make changes to your forked buildpack. If you are using the offline version of the

buildpack, you cannot override the version of the agent currently in use by an application. The buildpack bundles the dependencies.

The Contrast Security agent framework downloads the latest Contrast agent and creates a configuration file. The buildpack's detect script prints tags to standard output.

Configuration

The detect script confirms the existence of a single, bound Contrast service. A Contrast service exists if the `VCAP_SERVICES` payload contains a service name, label, or tag with `contrast-security` as a substring.

To bind Contrast with a user-provided service, you must have a name or tag with `contrast-security` in it. The credential payload must also contain the [standard YAML properties \(page 107\)](#).

This example creates a user-provided service and binds it to an application:

```
cf create-user-provided-service contrast-security-service -p
"teamserver_url, username, api_key, service_key"
cf bind-service spring-music contrast-security-service
cf restage spring-music
```



NOTE

The `teamserver_url` should be only protocol and hostname. Do not include `/` `Contrast/` or `/Contrast/api`.

See also

[Add Contrast service broker tile for VMware Tanzu \(page 346\)](#)

[Add Contrast service broker for VMware Tanzu \(page 348\)](#) to use the service broker without the tile.

Add the Contrast service broker tile for Node.js

With a service broker, VMware Tanzu (formerly Pivotal Cloud Foundry) applications can easily bind to and consume services from the Apps Manager or the command line. You can deploy the Contrast service broker as a Node.js application on VMware Tanzu, and use one or more Contrast accounts. The broker exposes the Contrast service on the VMware Tanzu marketplace so you can create a service instance.

When you add a tile, it creates one organization: the **contrast-security-service-broker-org**. Use this organization to deploy the Contrast service broker application. This requires 512MB of memory.

Before you begin

Before you add the Contrast service broker tile, you must have:

- Pivotal Apps Manager and Ops Manager
- An active Contrast account
- The default Node.js buildpack for any application using Contrast. If you have a custom buildpack, you must copy the Contrast framework support and configuration into it.

Steps

To add the Contrast service broker tile for Node.js:

1. Download the Contrast service broker tile from the [VMware Tanzu Network](#).

2. Select **Import a Product** and then select the *contrast-security-service-broker-#.#.#.pivotal* tile you downloaded.

**NOTE**

If the file you downloaded has a ZIP extension rename it to *contrast-security-service-broker-#.#.#.pivotal*.

3. To add a service plan select **Service Plans** in the Contrast service broker tile and select **Add**. The tile requires some configuration before you can deploy it. The service broker does not include service plans by default. You must add at least one before you can deploy the Contrast service broker tile.
4. Complete these configuration parameters in the service plan:
 - **TeamServer**: The URL for your Contrast application instance
 - **TeamServer Service Key**: [Organization service key \(page 104\)](#)
 - **TeamServer API Key**: [Organization API key \(page 104\)](#)
 - **Organization UUID**: [Organization ID \(page 104\)](#) to which the application will belong
 - **Username**: Your Contrast username
 - **Plan Name**: Name of the plan as it will appear in Apps Manager
 - **Proxy Host**: The hostname of a proxy for the service broker to communicate with Contrast
 - **Proxy Port**: The proxy port
 - **Proxy Username**: The proxy username if it requires authentication
 - **Plan Password**: The proxy password
5. After you define the service plan, select **Save**. If you want some applications to belong to different organizations, define the other plans you will need.
6. Select **Apply Changes** in the dashboard. This may take some time to finish.
7. Now, use the Cloud Foundry CLI to bind your application.
8. Clone the project to a local directory. For example: `Node/pcf/node-hello-world`.
9. Update the applications `package.json` to add the Contrast Node agent as a dependency. For example: `"@contrast/agent": "^5"`.
10. Update the start script in `package.json` to instrument with the agent. For example: `"start": "node --import @contrast/agent app.js"`.
11. Push the application to VMware Tanzu. For example, from the `Node/pcf` directory, run:

```
cf push myAppNodeBroker -p node-hello-world \
  -b 'https://github.com/cloudfoundry/nodejs-buildpack.git' \
  -t 180
```

12. Then run the following:

```
cf service-access
```

Example output:

```
Getting service access as admin...
broker: contrast-security-service-broker
  service      plan      access  orgs
contrast-security  apptwo  all
```

13. Then run the following:

```
cf create-service contrast-security apptwo contrast
```

Example output:

```
Creating service instance contrast in org system / space apps as admin...  
OK
```

14. Then run the following:

```
cf services
```

Example output (notice there are currently no bound apps):

```
Getting services in org system / space apps as admin...  
  
name           service           plan    bound apps    last operation  
broker                                     upgrade available  
contrast      contrast-security  apptwo                                     create succeeded  
contrast-security-service-broker
```

15. Run the following to bind the sample app to the service:

```
cf bind-service myAppNodeBroker contrast
```

Example output:

```
Binding service contrast to app myAppNodeBroker in org system / space  
apps as admin...  
OK
```

16. Run the following again to confirm the app is now bound to the service:

```
cf services
```

Example output:

```
Getting services in org system / space apps as admin...  
  
name           service           plan    bound apps    last  
operation      broker                                     upgrade available  
contrast      contrast-security  apptwo  myAppNodeBroker  create  
succeeded     contrast-security-service-broker
```

17. Run the following command to restage the application now that it is bound to the service:

```
cf restage myAppNodeBroker
```

18. Go to Contrast to view the application.

See also

[Add Contrast service broker \(page 348\)](#)

Add the Contrast service broker for Node.js

Use the Contrast service broker to easily bind services to an application in VMware Tanzu (Pivotal Cloud Foundry) and use the Contrast Node.js agent.

Steps

To set up VMware Tanzu:

1. [Contact Support](#).
2. Once you have a service broker source code, deploy the service broker application:

```
cf push contrast-security-service-broker
```

The service broker now appears in PCF.

3. Configure plans with the `CONTRAST_SERVICE_PLANS` environment variable (the service broker does not offer any plans by default).

You can also use the **Pivotal Ops Manager** to set the environment variables. If you are using IBM Cloud, you can select the application, select **Runtime** and then **Environment Variables** to set the value.

Example: This example shows how to set the value in the command line.

```
cf set-env contrast-security-service-broker CONTRAST_SERVICE_PLANS
" {
  "ServicePlan1": {
    "name": "ServicePlan1",
    "teamserver_url": "https://yourteamserverurl.com",
    "username": "your_username",
    "org_uuid": "00000000-1111-2222-3333-000000000000",
    "api_key": "your_api_key",
    "service_key": "your_service_key"
  },
  "AnotherServicePlan": {
    "name": "AnotherServicePlan",
    "teamserver_url": "https://yourteamserverurl.com",
    "username": "your_username",
    "org_uuid": "00000000-1111-2222-3333-000000000001",
    "api_key": "your_api_key",
    "service_key": "some_other_service_key"
  }
}
```

To run the agent on IBM Cloud, you must use single quotes to set the `CONTRAST_SERVICE_PLANS` environment variable. Example:

```
cf set-env contrast-security-service-broker CONTRAST_SERVICE_PLANS
" {
  'ServicePlan1': {
    'name': 'ServicePlan1',
    'teamserver_url': 'https://yourteamserverurl.com',
    'username': 'your_username',
    'org_uuid': '00000000-1111-2222-3333-000000000000',
    'api_key': 'your_api_key',
    'service_key': 'your_service_key'
  },
  'AnotherServicePlan': {
    'name': 'AnotherServicePlan',
    'teamserver_url': 'https://yourteamserverurl.com',
    'username': 'your_username',
    'org_uuid': '00000000-1111-2222-3333-000000000000',
    'api_key': 'your_api_key',
    'service_key': 'some_other_service_key'
  }
}
```

4. Restage your application with a command similar to this example:

```
cf restage contrast-security-service-broker
```

5. Set an environment variable for a username and a password:

```
cf set-env contrast-security-service-broker SECURITY_USER_NAME
aSecureUsername
cf set-env contrast-security-service-broker SECURITY_USER_PASSWORD
aSecurePassword
```

6. Create a service broker instance. Define at least one service plan for this. You must use the same username and password created in the previous step.

```
cf create-service-broker contrast-security-service-broker USER_NAME  
PASSWORD  
<URL of your application>
```

On IBM Cloud, add `--space-scoped` at the end of the command. For example:

```
cf create-service-broker contrast-security-service-broker USER_NAME  
PASSWORD  
<URL of your application> --space-scoped
```

7. All service brokers start as private. Make it public with:

```
cf enable-service-access contrast-security-service-broker
```

8. Once the service broker is working, create a service instance and bind it to the application. To create a service instance, run the following command:

```
cf create-service contrast-security-service-broker ServicePlan1  
<name_of_service>
```

9. Bind the service broker to your application by running the following command:

```
cf bind-service <app_name> <name_of_service>
```

You should now see the agent start up with your application. You will also see your application in Contrast.

See also

[Add Contrast service broker tile \(page 346\)](#)

Update the Node.js agent

The most reliable and effective way to automatically update the Contrast Node.js agent is to use the Node.js npm package manager to install and download the latest version available.

Because npm manages all dependencies for your Node.js application, it should already be available and part of your build environment. How frequently you update the Contrast Node.js agent and where you get updates depends on your organization's preferences and your Contrast implementation: hosted (SaaS) or on-premises (EOP).

You can either update the agent automatically or manually.

Before you begin

Before you begin, you should have:

- Some familiarity with DevOps practices and Node's npm package manager.
- Access to the npm repository for the Contrast agent.
- Confirmed that your Node.js application runs properly without the Contrast Node.js agent.
- Previously successfully installed the Contrast Node.js agent.
- Defined a policy for how and when to update the agent, based on your change management policy and the environment where you deploy agents.



IMPORTANT

Unless Contrast Support advises you to do so, do not use a version of the Contrast Node.js agent that is ahead of the version available from your Contrast instance.

Steps

1. You will install the Node.js agent from the npm public (or private) repository. Depending on your Contrast installation, you can use one or both sources to get the latest Contrast Node.js agent:
 - **Hosted (SaaS) installations:** You can get the latest version of the agent from npm. If your organization prefers to validate agents before using them, you can also use a private npm repository with approved versions only.
 - **On-premises (EOP) installations:** Many organizations that use on-premises installations do not immediately update core software or agents when Contrast releases new software. Public repositories (like npm) typically host new versions of the agent that are not designed or tested to work with older versions of Contrast. On-premises users should source agent updates from a private npm repository where you only store versions of the agent that match your on-premises Contrast installation.
2. Install the agent and use scripts for automatic updates using the best method for you:
 - **Use `package.json`:** This file specifies which dependencies will automatically resolve every time your Node.js application builds with artifacts from npm (public or private). Include the Contrast Node.js agent here to easily keep every new build of your application aligned with the latest version of the agent. For example:

```
{
  "name": "sample_application",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "start": "nodemon",
    "contrast": "node --import @contrast/agent index.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "express": "^4.17.1",
    "@contrast/agent": "latest",
  },
  "devDependencies": {
    "nodemon": "^1.19.2"
  }
}
```

Then use the `$ npm update` command whenever you build your application. This will automatically download, and add or update, the Contrast Node.js agent from npm to the Node.js application.

- **Install and update manually using command line:** For some organizations, the `package.json` file must be consistent across environments, or they do not plan to install the Contrast Node.js agent into all environments. In these cases, install the agent manually. You can manually update agents as part of a Node.js build process. Use this command to manually retrieve and add or update the Contrast Node.js agent from npm (public or private) to the Node.js application:

```
$ npm install @contrast/agent
```

3. To check whether the installation/update succeeded, run the following command and look for output similar to the following example:

```
npm ls --all | grep contrast
@contrast/agent@5.11.0
```

```
@contrast/agentify@1.28.0
@contrast/common@1.22.0
@contrast/config@1.29.0
@contrast/common@1.22.0 deduped
@contrast/core@1.33.0
@contrast/common@1.22.0 deduped
@contrast/find-package-json@1.2.0 deduped
@contrast/fn-inspect@4.2.0 deduped
@contrast/deadzones@1.3.0
@contrast/common@1.22.0 deduped
```

See also

- [Node.js supported technologies \(page 332\)](#)
- [Install Node.js \(page 334\)](#)

Configure the Node.js agent

The [standard configuration \(page 52\)](#) for all agents uses this [order of precedence \(page 106\)](#).

There are several ways to configure the Node.js agent, but generally, you should:

- Use [environment variables \(page 352\)](#) for agent authentication keys and application-specific configuration values. Learn more about [environment variables in general \(page 110\)](#).
- Use a YAML configuration file to set common configuration values for all applications in an organization or container (for example, to redirect logging or proxy configuration). This [template \(page 353\)](#) shows all valid configuration options for the Node.js agent. Learn more about [YAML configuration \(page 107\)](#) in general.



TIP

Use the [Contrast agent configuration editor \(page 109\)](#) to create or upload a YAML configuration file, validate YAML, and get setting recommendations. The editor also provides the correct environment variables, if desired.

Environment variables

Use environment variables for application-specific configuration values (like configuring server environment, application names or agent logging). You can also use environment variables to set any other valid properties for the Node.js agent.

You can see a full list of valid properties in the [Node.js YAML template \(page 353\)](#), but here are some common examples as environment variables:

Agent Token: This variable is a base64 encoded JSON object containing the url, api_key, service_key, and user_name configuration settings, allowing you to set them in a single variable.

- CONTRAST__API__TOKEN

If your agent configuration refers to both the legacy settings and the agent token, (in environment variables or the YAML file), the legacy settings take precedence. To use just the agent token value, remove references to the legacy settings.

Legacy settings: If you are using a Node.js agent version earlier than 5.15.0, the required variables are:

Environment variable	Description
CONTRAST__API__SERVICE_KEY	Set the service key needed to communicate with Contrast.
CONTRAST__API__API_KEY	Set the API key needed to communicate with Contrast.
CONTRAST__API__USER_NAME	Set the user name needed to communicate with Contrast.
CONTRAST__API__URL	Set the URL for the Contrast web interface.

Optional variables include:

Environment variable	Description
CONTRAST__APPLICATION_NAME	Override the reported application name.
CONTRAST__CONFIG_PATH	When set, supersedes the default location of the YAML configuration file. (Unlike other environment variables, this one cannot be set as a YAML property, and contains only single underscores.)
CONTRAST__SERVER_PATH	Override the reported server path.
CONTRAST__SERVER_NAME	Provides a consistent server name for cases where containerized apps generate many server records. This could be the microservice name or app name.
CONTRAST__AGENT_DIAGNOSTICS_ENABLE	Creates configuration and system files at startup to help track diagnostic and troubleshooting information. Default is <code>true</code> .
CONTRAST__AGENT_LOGGING_APPEND	When set to <code>false</code> , creates a new log file on startup instead of appending and rolling daily. Default is <code>true</code> .
CONTRAST__AGENT_LOGGING_LEVEL	Logging level: <code>FATAL</code> , <code>ERROR</code> , <code>WARN</code> , <code>INFO</code> , <code>DEBUG</code> or <code>TRACE</code> . Default is <code>ERROR</code> .
CONTRAST__AGENT_LOGGING_PATH	Where Contrast will put its debug log. Default is <code>node-contrast.log</code> .
CONTRAST__AGENT_LOGGING_STDOUT	When set to <code>false</code> , suppresses output to <code>stdout</code> . Default is <code>true</code> .

If you want to redirect logging for the Node.js, [contact Support](#) for assistance.



NOTE

For legacy v4 of the agent:

For the Node.js agent you must manually configure `DEBUG`. `INFO`-level statements aren't logged to the console unless the environment variable `DEBUG` is set to include the Contrast namespace: `DEBUG=contrast:*`. This could be useful in environments where you don't have access to the file system (like Docker or ECS).

If you want to redirect logging for the Node.js, see more examples on the [npm site](#) or [contact Support](#) for assistance.

Node.js YAML template

Use this template to configure the Node.js agent using a YAML configuration file. (Learn more about [YAML configuration \(page 107\)](#).)

Place your YAML file in the default location: `/etc/contrast/contrast_security.yaml`

```
#
=====
```

```
==
# Use the properties in this YAML file to configure a Contrast agent.
# Go to https://docs.contrastsecurity.com/en/order-of-precedence.html
# to determine the order of precedence for configuration values.
#
=====
==

# Use this setting if you want to temporarily disable a Contrast agent.
# Set to `true` to enable the agent; set to `false` to disable the agent.
# enable: true

#
=====
==
# api
# Use the properties in this section to connect the agent to the Contrast
# UI.
#
=====
==
api:

  # ***** REQUIRED *****
  # Set the URL for the Contrast UI.
  url: https://app.contrastsecurity.com/Contrast

  # ***** REQUIRED *****
  # Set the API key needed to communicate with the Contrast UI.
  api_key: NEEDS_TO_BE_SET

  # ***** REQUIRED *****
  # Set the service key needed to communicate with the Contrast
  # UI. It is used to calculate the Authorization header.
  service_key: NEEDS_TO_BE_SET

  # ***** REQUIRED *****
  # Set the user name used to communicate with the Contrast
  # UI. It is used to calculate the Authorization header.
  user_name: NEEDS_TO_BE_SET

  # base64 encoded JSON object containing the `url`,
  # `api_key`, `service_key`, and `user_name` config options,
  # allowing them all to be set in a single variable.
  # token: NEEDS_TO_BE_SET

  # Set the default request timeout.
  # timeout_ms: NEEDS_TO_BE_SET

  #
=====
# api.certificate
# Use the following properties for communication
# with the Contrast UI using certificates.
```

```
#
=====
# certificate:

# If set to `false`, the agent will ignore the
# certificate configuration in this section.
# enable: true

# Set the absolute or relative path to a CA for communication
# with the Contrast UI using a self-signed certificate.
# ca_file: NEEDS_TO_BE_SET

# Set the absolute or relative path to the Certificate
# PEM file for communication with the Contrast UI.
# cert_file: NEEDS_TO_BE_SET

# Set the absolute or relative path to the Key PEM
# file for communication with the Contrast UI.
# key_file: NEEDS_TO_BE_SET

# If the Key file requires a password, it can be set here or in
# the matching ENV value (`CONTRAST__CERTIFICATE__KEY_PASSWORD`).
# key_password: NEEDS_TO_BE_SET

#
=====
# api.proxy
# Use the following properties for communication
# with the Contrast UI over a proxy.
#
=====
# proxy:

# Set value to `true` for the agent to communicate
# with the Contrast web interface over a proxy. Set
# value to `false` if you don't want to use the proxy.
# enable: NEEDS_TO_BE_SET

# Set the URL for your Proxy Server. The URL form is `scheme://
host:port`.
# url: NEEDS_TO_BE_SET

#
=====
==
# agent
# Use the properties in this section to control the way and frequency
# with which the agent communicates to logs and the Contrast UI.
#
=====
==
# agent:

# Set to limit the length of Error stack traces to a specified number.
# Larger limits will improve accuracy but increase memory usage.
```

```
# stack_trace_limit: 10

#
=====
# agent.diagnostics
# Use the properties in this section to specify the information the agent
# should collect and report in order to diagnose problems in the agent.
#
#
=====
# diagnostics:

# Creates config and system info files
# at startup if true. True by default.
#
# The same thing can be achieved by setting the
# CONTRAST__AGENT__DIAGNOSTICS__ENABLE=[true/false] env variable.
#
# enable: true

# Set the directory in which to write diagnostic files.
# Defaults to the application's current working directory.
# report_path: ./

#
=====
# agent.route_coverage
# Use the following properties for the route-based coverage feature.
#
#
=====
# route_coverage: {}

#
=====
# agent.logger
# Define the following properties to set logging values.
# If the following properties are not defined, the
# agent uses the logging values from the Contrast UI.
#
#
=====
# logger:

# Enable diagnostic logging by setting a path to a log file.
# While diagnostic logging hurts performance, it generates
# useful information for debugging Contrast. The value set here
# is the location to which the agent saves log output. If no
# log file exists at this location, the agent creates a file.
#
# Example - `/opt/Contrast/contrast.log` creates a log in the
# `/opt/Contrast` directory, and rotates it automatically as needed.
#
# path: ./contrast_agent.log

# Set the the log output level. Valid options are
# `ERROR`, `WARN`, `INFO`, `DEBUG`, and `TRACE`.
```



```
# level: INFO

# Set to `false` for the agent to always create a
# new log file instead of appending and rolling.
# append: true

# Set to `false` to suppress log output to `stdout`.
# stdout: true

# Set the roll size for log files in megabytes. The agent will
# attempt to prevent the log file from being larger than this size.
# This feature is only available in agent version >=4.0.0
# roll_size: 100M

# Set the number of backup files to keep. Set to `0` to disable.
# This feature is only available in agent version >=4.0.0
# backups: 10

#
=====
# agent.security_logger
# Define the following properties to set security logging
# values associated with Protect. If not defined, the agent
# uses the security logging (CEF) values from the Contrast UI.
#
=====
# security_logger:

# Set the file to which the agent logs security events.
# path: ./contrast/security.log

# Set the log level for security logging. Valid options
# are `ERROR`, `WARN`, `INFO`, `DEBUG`, and `TRACE`.
# level: ERROR

# Set to `true` to log output to `stdout` as well as the configured
file.
# stdout: false

#
=====
# agent.security_logger.syslog
# Define the following properties to set Syslog values. If the
properties
# are not defined, the agent uses the Syslog values from the Contrast
UI.
#
=====
# syslog:

# Set to `true` to enable Syslog logging.
# enable: NEEDS_TO_BE_SET

# Set the IP address of the Syslog server
# to which the agent should send messages.
```

```
# ip: NEEDS_TO_BE_SET

# Set the port of the Syslog server to
# which the agent should send messages.
# port: NEEDS_TO_BE_SET

# Set the facility code of the messages the agent sends to Syslog.
# facility: 19

# Set the log level of Exploited attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_exploited: ALERT

# Set the log level of Blocked attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_blocked: NOTICE

# Set the log level of Blocked At Perimeter
# attacks. Value options are `ALERT`, `CRITICAL`,
# `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_blocked_perimeter: NOTICE

# Set the log level of Probed attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_probed: WARNING

# Set the log level of Suspicious attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_suspicious: WARNING

#
=====
# agent.service
# The following properties are used by the Contrast Service.
#
=====
# service:

# Set to `false` to disallow the service to be started, and
# effectively disable the agent, if read by the service. If the
# agent reads this property, it disallows service auto-start.
# enable: true

# Set to `true` to enable listening for gRPC connections.
# The `socket`, `host` and `port` fields will be used for
# configuring the gRPC server in place of the legacy RPC server.
# grpc: false

# If this property is defined, the service is
# listening on a Unix socket at the defined path.
# socket: /tmp/service.sock

# ***** REQUIRED *****
# Set the the hostname or IP address of the Contrast
# service to which the Contrast agent should report.
```

```
host: localhost

# ***** REQUIRED *****
# Set the the port of the Contrast service
# to which the Contrast agent should report.
port: 30555

#
=====
# agent.service.teamserver_retry
# The following properties are used by the Teamserver HTTP client
# to configure failed request retrying in the Contrast service.
#
=====
# teamserver_retry:

# Enable retrying HTTP requests to the Teamserver endpoint.
# enable: true

# How long to wait between retries in milliseconds.
# interval_ms: 5000

# How many times to retry HTTP requests to Teamserver before giving
up.
# max_attempts: 3

#
=====
# agent.service.logger
# The following properties are used by the logger in the
# Contrast service. If the properties are not defined, the
# service uses the logging values from the Contrast UI.
#
=====
# logger:

# Set the location to which the Contrast service saves log output.
# If no log file exists at this location, the service creates one.
#
# Example - `/opt/Contrast/contrast_service.log` will
# create a log in the `/opt/Contrast` directory.
#
# path: ./contrast_service.log

# Set the the log output level. Options are `OFF`, `FATAL`,
# `ERROR`, `WARN`, `INFO`, `DEBUG`, `TRACE`, and `ALL`.
# level: ERROR

# Override the name of the process used in logs.
# progname: Contrast Service

# Set to `true` to send log output to `stdout`.
# stdout: false

#
```

```
=====
# agent.heap_dump
# The following properties are used to trigger heap dumps from within
# the agent to snapshot the behavior of instrumented applications.
#
=====
# heap_dump:

# Set to `true` for the agent to automatically
# take heap dumps of the instrumented application.
# enable: false

# Set the location to which to save the heap dump files. If relative,
# the path is determined based on the process' working directory.
# path: contrast_heap_dumps

# Set the amount of time to wait, in milliseconds,
# after agent startup to begin taking heap dumps.
# delay_ms: 10_000

# Set the amount of time to wait, in milliseconds, between each heap
dump.
# window_ms: 10_000

# Set the number of heap dumps to take before disabling this feature.
# count: 5

#
=====
# agent.node
# The following properties apply to any Node configurations.
#
=====
# node:

# Set the directory containing the application's `package.json` file.
# app_root: NEEDS_TO_BE_SET

#
=====
# agent.node.rewrite
# Use the following properties to set up source code rewriting.
#
=====
# rewrite:

#
=====
# agent.node.rewrite.cache
# Use the following properties to set up rewrite caching.
#
=====
# cache:

# Set to `false` to disable caching rewritten source code files.
```

```
# enable: true

# Set the directory in which to cache rewritten
# source code files. Defaults to `.contrast/` in
# the application's current working directory.
# path: .contrast

#
=====
# agent.node.source_maps
# Use the following properties to set up
# source map generation when rewriting.
#
=====
# source_maps:

# Set to `false` to disable source map generation when rewriting.
# enable: true

#
=====
# agent.node.library_usage
# Configuration for Node.js library usage reporting
#
=====
# library_usage:

#
=====
# agent.node.library_usage.reporting
# Use the following properties to set
# up enhanced library usage reporting.
#
=====
# reporting:

# Set to `false` to disable enhanced library usage features, i.e.
# scanning for composition of dependencies, reporting library usage.
# enable: true

# Set the interval (in milliseconds) for
# collecting code events for library usage.
# interval_ms: 1

#
=====
==
# inventory
# Use the properties in this section to override the inventory features.
#
=====
==
# inventory:

# Set to `false` to disable library analysis.
```

```
# analyze_libraries: true

# Apply a list of labels to libraries. Labels
# must be formatted as a comma-delimited list.
# Example - `label1, label2, label3`
#
# tags: NEEDS_TO_BE_SET

#
=====
==
# assess
# Use the properties in this section to control Assess.
#
=====
==
# assess:

# Include this property to determine if the Assess
# feature should be enabled. If this property is not
# present, the decision is delegated to the Contrast UI.
# enable: false

# Apply a list of labels to vulnerabilities and preflight
# messages. Labels must be formatted as a comma-delimited list.
# Example - `label1, label2, label3`
#
# tags: NEEDS_TO_BE_SET

# Value options are `ALL`, `SOME`, or `NONE`.
# stacktraces: ALL

# Set to `true` to trust incoming strings when they pass custom
# validators (Mongoose, Joi, validator, fastify-static).
# This feature is only available in agent version 4.10.0 and later
# trust_custom_validators: false

# Enables the serve-static module as a path-traversal
# sanitizer. Express uses serve-static in a safe way
# but manual setup of serve-static can be vulnerable.
#
# Even with Express there is a possibility for "traversing-down" the
served
# folder or user misconfiguration if not configured with an absolute path
#
# This feature is only available in agent version 4.31.0 and later
# enable_sanitizer_serve_static: false

# When set to `true`, string tracking will occur lazily as user-controlled
# values are accessed by application code. When `false`, tracking will
# occur at the time of input parsing and will be limited to 250 values.
# This feature is only available in agent version 4.18.0 and later
# enable_lazy_tracking: true

#
```

```
=====
# assess.sampling
# Use the following properties to control sampling in the agent.
#
=====
# sampling:

# Set to `true` to enable sampling.
# enable: false

# This property indicates the number of requests
# to analyze in each window before sampling begins.
# baseline: 5

#
=====
# assess.probabilistic_sampling
# Use the following properties to control
# probabilistic sampling in the agent.
#
=====
# probabilistic_sampling:

# Set to `true` to enable probabilistic sampling.
# enable: false

# The probability that a request will be analyzed. Each
# request will independently share this same probability
# of being sampled. Value needs to be within range [0, 1].
# base_probability: 0.10

#
=====
==
# protect
# Use the properties in this section to override Protect features.
#
=====
==
# protect:

# Include this property to determine if the Protect
# feature should be enabled. If this property is not
# present, the decision is delegated to the Contrast UI.
# enable: false

#
=====
# protect.probe_analysis
# Use the settings in this section to
# control the behavior of probe analysis.
# Support for this option is limited to Node agent versions >= 5
#
=====
# probe_analysis:
```

```
# Set to `false` to disable probe analysis.
# enable: true

#
=====
# protect.rules
# Use the following properties to set simple rule configurations.
#
=====
# rules:

# Define a list of Protect rules to disable in the agent. To view a
# list of rule names, in Contrast go to user menu > Policy Management >
# Protect rules. The rules must be formatted as a comma-delimited list.
# disabled_rules: NEEDS_TO_BE_SET

#
=====
# protect.rules.bot-blocker
# Use the following selection to configure if the
# agent blocks bots. Set to `true` to enable blocking.
#
=====
# bot-blocker:

# Set to `true` for the agent to block known bots.
# enable: false

#
=====
# protect.rules.sql-injection
# Use the following settings to configure the sql-injection rule.
#
=====
# sql-injection:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or off.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.cmd-injection
# Use the following properties to configure
# how the command injection rule works.
#
=====
# cmd-injection:

# Set the mode of the rule. Value options are
```



```
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.cmd-injection-semantic-chained-commands
# Use the following properties to configure how the
# 'command injection - chained commands' rule works
#
=====
# cmd-injection-semantic-chained-commands:

# Set the mode of the rule. Value options
# are `monitor`, `block`, or `off`.
# mode: off

#
=====
# protect.rules.cmd-injection-semantic-dangerous-paths
# Use the following properties to configure how the
# 'command injection - dangerous paths' rule works
#
=====
# cmd-injection-semantic-dangerous-paths:

# Set the mode of the rule. Value options
# are `monitor`, `block`, or `off`.
# mode: off

#
=====
# protect.rules.cmd-injection-command-backdoors
# Use the following properties to configure how the
# 'command injection - command backdoors' rule works
#
=====
# cmd-injection-command-backdoors:

# Set the mode of the rule. Value options
# are `monitor`, `block`, or `off`.
# mode: off

#
=====
# protect.rules.path-traversal-semantic-file-security-bypass
# Use the following properties to configure how the
# 'path traversal - file security bypass' rule works
#
=====
# path-traversal-semantic-file-security-bypass:
```

```
# Set the mode of the rule. Value options
# are `monitor`, `block`, or `off`.
# mode: off

#
=====
# protect.rules.path-traversal
# Use the following properties to configure
# how the path traversal rule works.
#
=====
# path-traversal:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.method-tampering
# Use the following properties to configure
# how the method tampering rule works.
#
=====
# method-tampering:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.reflected-xss
# Use the following properties to configure how
# the reflected cross-site scripting rule works.
#
=====
# reflected-xss:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off
```

```
#
=====
# protect.rules.unsafe-file-upload
# Use the following properties to configure
# how the unsafe file upload rule works.
#
=====
# unsafe-file-upload:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.xxe
# Use the following properties to configure
# how the XML external entity works.
#
=====
# xxe:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.untrusted-deserialization
# Use the following properties to configure
# how the untrusted deserialization rule works.
#
=====
# untrusted-deserialization:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.ssjs-injection
# Use the following properties to configure
```

```
# how the SSJS Injection rule works.
#
=====
# ssjs-injection:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.nosql-injection
# Use the following properties to configure
# how the NOSQL Injection rule works.
#
=====
# nosql-injection:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.nosql-injection-mongo
# Use the following properties to configure
# how the NOSQL Injection rule works.
#
=====
# nosql-injection-mongo:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
==
# application
# Use the properties in this section for
# the application(s) hosting this agent.
#
=====
```

```
==
# application:

# Override the reported application name.
#
# Note - On systems where multiple, distinct applications may be served
# by a single process, this configuration causes the agent to report
# all discovered applications as one application with the given name.
#
# Note - On Java systems where multiple, distinct applications may be
# served by a single process, this configuration causes the agent to
report
# all discovered applications as one application with the given name.
#
# name: NEEDS_TO_BE_SET

# Override the reported application path.
# path: NEEDS_TO_BE_SET

# Add the name of the application group with which this
# application should be associated in the Contrast UI.
# group: NEEDS_TO_BE_SET

# Add the application code this application should use in the Contrast UI.
# code: NEEDS_TO_BE_SET

# Override the reported application version.
# version: NEEDS_TO_BE_SET

# Pass arguments to the underlying application.
# args: NEEDS_TO_BE_SET

# Apply labels to an application. Labels must
# be formatted as a comma-delimited list.
# Example - `label1,label2,label3`
#
# tags: NEEDS_TO_BE_SET

# Define a set of `key=value` pairs (which conforms to RFC 2253) for
# specifying user-defined metadata associated with the application. The
# set must be formatted as a comma-delimited list of `key=value` pairs.
# Example - `business-unit=accounting, office=Baltimore`
#
# metadata: NEEDS_TO_BE_SET

# Provide the ID of a session which already exists in the Contrast
# UI. Vulnerabilities discovered by the agent are associated with
# this session. If an invalid ID is supplied, the agent will be
# disabled. This option and `application.session_metadata` are
# mutually exclusive; if both are set, the agent will be disabled.
# session_id: NEEDS_TO_BE_SET

# Provide metadata which is used to create a new session ID in the
# Contrast UI. Vulnerabilities discovered by the agent are associated with
# this new session. This value should be formatted as `key=value` pairs
```

```
# (conforming to RFC 2253). Available key names for this configuration
# are branchName, buildNumber, commitHash, committer, gitTag, repository,
# testRun, and version. This option and `application.session_id` are
# mutually exclusive; if both are set the agent will be disabled.
# session_metadata: NEEDS_TO_BE_SET

#
=====
==
# server
# Use the settings in this section to set metadata for the server
# hosting this agent. Contrast recognizes common, supported server
# names, paths, types and environments. Doing this may require a new
# server or license, and it may affect functionality of some features.
#
=====
==
# server:

# Override the reported server name.
# name: localhost

# Override the reported server path.
# path: NEEDS_TO_BE_SET

# Override the reported server type.
# type: NEEDS_TO_BE_SET

# Set the environment directly to override the default set
# by the Contrast UI. This allows the user to configure the
# environment dynamically at startup rather than manually
# updating the Server in the Contrast UI themselves afterwards.
#
# Valid values include `QA`, `PRODUCTION` and `DEVELOPMENT`.
# For example, `PRODUCTION` registers this Server as
# running in a `PRODUCTION` environment, regardless of the
# organization's default environment in the Contrast UI.
#
# environment: NEEDS_TO_BE_SET

# Apply a list of labels to the server. Labels
# must be formatted as a comma-delimited list.
# Example - `label1,label2,label3`
#
# tags: NEEDS_TO_BE_SET

# Set to `false` to disable detection of cloud
# provider metadata such as resource identifiers.
# discover_cloud_resource: true

#
=====
==
# Use the properties in this YAML file to configure a Contrast agent.
# Go to https://docs.contrastsecurity.com/en/order-of-precedence.html
```

```
# to determine the order of precedence for configuration values.
#
=====
==

# Use this setting if you want to temporarily disable a Contrast agent.
# Set to `true` to enable the agent; set to `false` to disable the agent.
# enable: true

#
=====
==
# api
# Use the properties in this section to connect the agent to the Contrast
UI.
#
=====
==
api:

# ***** REQUIRED *****
# Set the URL for the Contrast UI.
url: https://app.contrastsecurity.com/Contrast

# ***** REQUIRED *****
# Set the API key needed to communicate with the Contrast UI.
api_key: NEEDS_TO_BE_SET

# ***** REQUIRED *****
# Set the service key needed to communicate with the Contrast
# UI. It is used to calculate the Authorization header.
service_key: NEEDS_TO_BE_SET

# ***** REQUIRED *****
# Set the user name used to communicate with the Contrast
# UI. It is used to calculate the Authorization header.
user_name: NEEDS_TO_BE_SET

# base64 encoded JSON object containing the `url`,
# `api_key`, `service_key`, and `user_name` config options,
# allowing them all to be set in a single variable.
# token: NEEDS_TO_BE_SET

# Set the default request timeout.
# timeout_ms: NEEDS_TO_BE_SET

#
=====
# api.certificate
# Use the following properties for communication
# with the Contrast UI using certificates.
#
=====
# certificate:
```

```
# If set to `false`, the agent will ignore the
# certificate configuration in this section.
# enable: true

# Set the absolute or relative path to a CA for communication
# with the Contrast UI using a self-signed certificate.
# ca_file: NEEDS_TO_BE_SET

# Set the absolute or relative path to the Certificate
# PEM file for communication with the Contrast UI.
# cert_file: NEEDS_TO_BE_SET

# Set the absolute or relative path to the Key PEM
# file for communication with the Contrast UI.
# key_file: NEEDS_TO_BE_SET

# If the Key file requires a password, it can be set here or in
# the matching ENV value (`CONTRAST__CERTIFICATE__KEY_PASSWORD`).
# key_password: NEEDS_TO_BE_SET

#
=====
# api.proxy
# Use the following properties for communication
# with the Contrast UI over a proxy.
#
=====
# proxy:

# Set value to `true` for the agent to communicate
# with the Contrast web interface over a proxy. Set
# value to `false` if you don't want to use the proxy.
# enable: NEEDS_TO_BE_SET

# Set the URL for your Proxy Server. The URL form is `scheme://
host:port`.
# url: NEEDS_TO_BE_SET

#
=====
==
# agent
# Use the properties in this section to control the way and frequency
# with which the agent communicates to logs and the Contrast UI.
#
=====
==
# agent:

# Set to limit the length of Error stack traces to a specified number.
# Larger limits will improve accuracy but increase memory usage.
# stack_trace_limit: 10

#
```



```
=====
# agent.diagnostics
# Use the properties in this section to specify the information the agent
# should collect and report in order to diagnose problems in the agent.
#
#
=====
# diagnostics:

# Creates config and system info files
# at startup if true. True by default.
#
# The same thing can be achieved by setting the
# CONTRAST__AGENT__DIAGNOSTICS__ENABLE=[true/false] env variable.
#
# enable: true

# Set the directory in which to write diagnostic files.
# Defaults to the application's current working directory.
# report_path: ./

#
=====
# agent.route_coverage
# Use the following properties for the route-based coverage feature.
#
=====
# route_coverage: {}

#
=====
# agent.logger
# Define the following properties to set logging values.
# If the following properties are not defined, the
# agent uses the logging values from the Contrast UI.
#
=====
# logger:

# Enable diagnostic logging by setting a path to a log file.
# While diagnostic logging hurts performance, it generates
# useful information for debugging Contrast. The value set here
# is the location to which the agent saves log output. If no
# log file exists at this location, the agent creates a file.
#
# Example - `/opt/Contrast/contrast.log` creates a log in the
# `/opt/Contrast` directory, and rotates it automatically as needed.
#
# path: ./contrast_agent.log

# Set the the log output level. Valid options are
# `ERROR`, `WARN`, `INFO`, `DEBUG`, and `TRACE`.
# level: INFO

# Set to `false` for the agent to always create a
```

```
# new log file instead of appending and rolling.
# append: true

# Set to `false` to suppress log output to `stdout`.
# stdout: true

# Set the roll size for log files in megabytes. The agent will
# attempt to prevent the log file from being larger than this size.
# This feature is only available in agent version >=4.0.0
# roll_size: 100M

# Set the number of backup files to keep. Set to `0` to disable.
# This feature is only available in agent version >=4.0.0
# backups: 10

#
=====
# agent.security_logger
# Define the following properties to set security logging
# values associated with Protect. If not defined, the agent
# uses the security logging (CEF) values from the Contrast UI.
#
=====
# security_logger:

# Set the file to which the agent logs security events.
# path: ./contrast/security.log

# Set the log level for security logging. Valid options
# are `ERROR`, `WARN`, `INFO`, `DEBUG`, and `TRACE`.
# level: ERROR

# Set to `true` to log output to `stdout` as well as the configured
file.
# stdout: false

#
=====
# agent.security_logger.syslog
# Define the following properties to set Syslog values. If the
properties
# are not defined, the agent uses the Syslog values from the Contrast
UI.
#
=====
# syslog:

# Set to `true` to enable Syslog logging.
# enable: NEEDS_TO_BE_SET

# Set the IP address of the Syslog server
# to which the agent should send messages.
# ip: NEEDS_TO_BE_SET

# Set the port of the Syslog server to
```

```
# which the agent should send messages.
# port: NEEDS_TO_BE_SET

# Set the facility code of the messages the agent sends to Syslog.
# facility: 19

# Set the log level of Exploited attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_exploited: ALERT

# Set the log level of Blocked attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_blocked: NOTICE

# Set the log level of Blocked At Perimeter
# attacks. Value options are `ALERT`, `CRITICAL`,
# `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_blocked_perimeter: NOTICE

# Set the log level of Probed attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_probed: WARNING

# Set the log level of Suspicious attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_suspicious: WARNING

#
=====
# agent.service
# The following properties are used by the Contrast Service.
#
=====
# service:

# Set to `false` to disallow the service to be started, and
# effectively disable the agent, if read by the service. If the
# agent reads this property, it disallows service auto-start.
# enable: true

# Set to `true` to enable listening for gRPC connections.
# The `socket`, `host` and `port` fields will be used for
# configuring the gRPC server in place of the legacy RPC server.
# grpc: false

# If this property is defined, the service is
# listening on a Unix socket at the defined path.
# socket: /tmp/service.sock

# ***** REQUIRED *****
# Set the the hostname or IP address of the Contrast
# service to which the Contrast agent should report.
host: localhost

# ***** REQUIRED *****
```

```
# Set the the port of the Contrast service
# to which the Contrast agent should report.
port: 30555

#
=====
# agent.service.teamserver_retry
# The following properties are used by the Teamserver HTTP client
# to configure failed request retrying in the Contrast service.
#
=====
# teamserver_retry:

# Enable retrying HTTP requests to the Teamserver endpoint.
# enable: true

# How long to wait between retries in milliseconds.
# interval_ms: 5000

# How many times to retry HTTP requests to Teamserver before giving
up.
# max_attempts: 3

#
=====
# agent.service.logger
# The following properties are used by the logger in the
# Contrast service. If the properties are not defined, the
# service uses the logging values from the Contrast UI.
#
=====
# logger:

# Set the location to which the Contrast service saves log output.
# If no log file exists at this location, the service creates one.
#
# Example - `/opt/Contrast/contrast_service.log` will
# create a log in the `/opt/Contrast` directory.
#
# path: ./contrast_service.log

# Set the the log output level. Options are `OFF`, `FATAL`,
# `ERROR`, `WARN`, `INFO`, `DEBUG`, `TRACE`, and `ALL`.
# level: ERROR

# Override the name of the process used in logs.
# progname: Contrast Service

# Set to `true` to send log output to `stdout`.
# stdout: false

#
=====
# agent.heap_dump
# The following properties are used to trigger heap dumps from within
```

```
# the agent to snapshot the behavior of instrumented applications.
#
=====
# heap_dump:

# Set to `true` for the agent to automatically
# take heap dumps of the instrumented application.
# enable: false

# Set the location to which to save the heap dump files. If relative,
# the path is determined based on the process' working directory.
# path: contrast_heap_dumps

# Set the amount of time to wait, in milliseconds,
# after agent startup to begin taking heap dumps.
# delay_ms: 10_000

# Set the amount of time to wait, in milliseconds, between each heap
dump.
# window_ms: 10_000

# Set the number of heap dumps to take before disabling this feature.
# count: 5

#
=====
# agent.node
# The following properties apply to any Node configurations.
#
=====
# node:

# Set the directory containing the application's `package.json` file.
# app_root: NEEDS_TO_BE_SET

#
=====
# agent.node.rewrite
# Use the following properties to set up source code rewriting.
#
=====
# rewrite:

#
=====
# agent.node.rewrite.cache
# Use the following properties to set up rewrite caching.
#
=====
# cache:

# Set to `false` to disable caching rewritten source code files.
# enable: true

# Set the directory in which to cache rewritten
```

```
# source code files. Defaults to `.contrast/` in
# the application's current working directory.
# path: .contrast

#
=====
# agent.node.source_maps
# Use the following properties to set up
# source map generation when rewriting.
#
=====
# source_maps:

# Set to `false` to disable source map generation when rewriting.
# enable: true

#
=====
# agent.node.library_usage
# Configuration for Node.js library usage reporting
#
=====
# library_usage:

#
=====
# agent.node.library_usage.reporting
# Use the following properties to set
# up enhanced library usage reporting.
#
=====
# reporting:

# Set to `false` to disable enhanced library usage features, i.e.
# scanning for composition of dependencies, reporting library usage.
# enable: true

# Set the interval (in milliseconds) for
# collecting code events for library usage.
# interval_ms: 1

#
=====
==
# inventory
# Use the properties in this section to override the inventory features.
#
=====
==
# inventory:

# Set to `false` to disable library analysis.
# analyze_libraries: true

# Apply a list of labels to libraries. Labels
```

```
# must be formatted as a comma-delimited list.
# Example - `label1, label2, label3`
#
# tags: NEEDS_TO_BE_SET

#
=====
==
# assess
# Use the properties in this section to control Assess.
#
=====
==
# assess:

# Include this property to determine if the Assess
# feature should be enabled. If this property is not
# present, the decision is delegated to the Contrast UI.
# enable: false

# Apply a list of labels to vulnerabilities and preflight
# messages. Labels must be formatted as a comma-delimited list.
# Example - `label1, label2, label3`
#
# tags: NEEDS_TO_BE_SET

# Value options are `ALL`, `SOME`, or `NONE`.
# stacktraces: ALL

# Set to `true` to trust incoming strings when they pass custom
# validators (Mongoose, Joi, validator, fastify-static).
# This feature is only available in agent version 4.10.0 and later
# trust_custom_validators: false

# Enables the serve-static module as a path-traversal
# sanitizer. Express uses serve-static in a safe way
# but manual setup of serve-static can be vulnerable.
#
# Even with Express there is a possibility for "traversing-down" the
served
# folder or user misconfiguration if not configured with an absolute path
#
# This feature is only available in agent version 4.31.0 and later
# enable_sanitizer_serve_static: false

# When set to `true`, string tracking will occur lazily as user-controlled
# values are accessed by application code. When `false`, tracking will
# occur at the time of input parsing and will be limited to 250 values.
# This feature is only available in agent version 4.18.0 and later
# enable_lazy_tracking: true

#
=====
# assess.sampling
# Use the following properties to control sampling in the agent.
```

```
#
=====
# sampling:

# Set to `true` to enable sampling.
# enable: false

# This property indicates the number of requests
# to analyze in each window before sampling begins.
# baseline: 5

#
=====
# assess.probabilistic_sampling
# Use the following properties to control
# probabilistic sampling in the agent.
#
=====
# probabilistic_sampling:

# Set to `true` to enable probabilistic sampling.
# enable: false

# The probability that a request will be analyzed. Each
# request will independently share this same probability
# of being sampled. Value needs to be within range [0, 1].
# base_probability: 0.10

#
=====
==
# protect
# Use the properties in this section to override Protect features.
#
=====
==
# protect:

# Include this property to determine if the Protect
# feature should be enabled. If this property is not
# present, the decision is delegated to the Contrast UI.
# enable: false

#
=====
# protect.probe_analysis
# Use the settings in this section to
# control the behavior of probe analysis.
# Support for this option is limited to Node agent versions >= 5
#
=====
# probe_analysis:

# Set to `false` to disable probe analysis.
# enable: true
```



```
#
=====
# protect.rules
# Use the following properties to set simple rule configurations.
#
=====
# rules:

# Define a list of Protect rules to disable in the agent. To view a
# list of rule names, in Contrast go to user menu > Policy Management >
# Protect rules. The rules must be formatted as a comma-delimited list.
# disabled_rules: NEEDS_TO_BE_SET

#
=====
# protect.rules.bot-blocker
# Use the following selection to configure if the
# agent blocks bots. Set to `true` to enable blocking.
#
=====
# bot-blocker:

# Set to `true` for the agent to block known bots.
# enable: false

#
=====
# protect.rules.sql-injection
# Use the following settings to configure the sql-injection rule.
#
=====
# sql-injection:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or off.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.cmd-injection
# Use the following properties to configure
# how the command injection rule works.
#
=====
# cmd-injection:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
```

```
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.cmd-injection-semantic-chained-commands
# Use the following properties to configure how the
# 'command injection - chained commands' rule works
#
=====
# cmd-injection-semantic-chained-commands:

# Set the mode of the rule. Value options
# are `monitor`, `block`, or `off`.
# mode: off

#
=====
# protect.rules.cmd-injection-semantic-dangerous-paths
# Use the following properties to configure how the
# 'command injection - dangerous paths' rule works
#
=====
# cmd-injection-semantic-dangerous-paths:

# Set the mode of the rule. Value options
# are `monitor`, `block`, or `off`.
# mode: off

#
=====
# protect.rules.cmd-injection-command-backdoors
# Use the following properties to configure how the
# 'command injection - command backdoors' rule works
#
=====
# cmd-injection-command-backdoors:

# Set the mode of the rule. Value options
# are `monitor`, `block`, or `off`.
# mode: off

#
=====
# protect.rules.path-traversal-semantic-file-security-bypass
# Use the following properties to configure how the
# 'path traversal - file security bypass' rule works
#
=====
# path-traversal-semantic-file-security-bypass:

# Set the mode of the rule. Value options
# are `monitor`, `block`, or `off`.
# mode: off
```

```
#
=====
# protect.rules.path-traversal
# Use the following properties to configure
# how the path traversal rule works.
#
=====
# path-traversal:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.method-tampering
# Use the following properties to configure
# how the method tampering rule works.
#
=====
# method-tampering:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.reflected-xss
# Use the following properties to configure how
# the reflected cross-site scripting rule works.
#
=====
# reflected-xss:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.unsafe-file-upload
```

```
# Use the following properties to configure
# how the unsafe file upload rule works.
#
=====
# unsafe-file-upload:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.xxe
# Use the following properties to configure
# how the XML external entity works.
#
=====
# xxe:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.untrusted-deserialization
# Use the following properties to configure
# how the untrusted deserialization rule works.
#
=====
# untrusted-deserialization:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.ssjs-injection
# Use the following properties to configure
# how the SSJS Injection rule works.
#
=====
```

```
# ssjs-injection:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.nosql-injection
# Use the following properties to configure
# how the NOSQL Injection rule works.
#
=====
# nosql-injection:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.nosql-injection-mongo
# Use the following properties to configure
# how the NOSQL Injection rule works.
#
=====
# nosql-injection-mongo:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
==
# application
# Use the properties in this section for
# the application(s) hosting this agent.
#
=====
==
# application:
```

```
# Override the reported application name.
#
# Note - On systems where multiple, distinct applications may be served
# by a single process, this configuration causes the agent to report
# all discovered applications as one application with the given name.
#
# Note - On Java systems where multiple, distinct applications may be
# served by a single process, this configuration causes the agent to
report
# all discovered applications as one application with the given name.
#
# name: NEEDS_TO_BE_SET

# Override the reported application path.
# path: NEEDS_TO_BE_SET

# Add the name of the application group with which this
# application should be associated in the Contrast UI.
# group: NEEDS_TO_BE_SET

# Add the application code this application should use in the Contrast UI.
# code: NEEDS_TO_BE_SET

# Override the reported application version.
# version: NEEDS_TO_BE_SET

# Pass arguments to the underlying application.
# args: NEEDS_TO_BE_SET

# Apply labels to an application. Labels must
# be formatted as a comma-delimited list.
# Example - `label1,label2,label3`
#
# tags: NEEDS_TO_BE_SET

# Define a set of `key=value` pairs (which conforms to RFC 2253) for
# specifying user-defined metadata associated with the application. The
# set must be formatted as a comma-delimited list of `key=value` pairs.
# Example - `business-unit=accounting, office=Baltimore`
#
# metadata: NEEDS_TO_BE_SET

# Provide the ID of a session which already exists in the Contrast
# UI. Vulnerabilities discovered by the agent are associated with
# this session. If an invalid ID is supplied, the agent will be
# disabled. This option and `application.session_metadata` are
# mutually exclusive; if both are set, the agent will be disabled.
# session_id: NEEDS_TO_BE_SET

# Provide metadata which is used to create a new session ID in the
# Contrast UI. Vulnerabilities discovered by the agent are associated with
# this new session. This value should be formatted as `key=value` pairs
# (conforming to RFC 2253). Available key names for this configuration
# are branchName, buildNumber, commitHash, committer, gitTag, repository,
# testRun, and version. This option and `application.session_id` are
```

```
# mutually exclusive; if both are set the agent will be disabled.
# session_metadata: NEEDS_TO_BE_SET

#
=====
==
# server
# Use the settings in this section to set metadata for the server
# hosting this agent. Contrast recognizes common, supported server
# names, paths, types and environments. Doing this may require a new
# server or license, and it may affect functionality of some features.
#
=====
==
# server:

# Override the reported server name.
# name: localhost

# Override the reported server path.
# path: NEEDS_TO_BE_SET

# Override the reported server type.
# type: NEEDS_TO_BE_SET

# Set the environment directly to override the default set
# by the Contrast UI. This allows the user to configure the
# environment dynamically at startup rather than manually
# updating the Server in the Contrast UI themselves afterwards.
#
# Valid values include `QA`, `PRODUCTION` and `DEVELOPMENT`.
# For example, `PRODUCTION` registers this Server as
# running in a `PRODUCTION` environment, regardless of the
# organization's default environment in the Contrast UI.
#
# environment: NEEDS_TO_BE_SET

# Apply a list of labels to the server. Labels
# must be formatted as a comma-delimited list.
# Example - `label1,label2,label3`
#
# tags: NEEDS_TO_BE_SET

# Set to `false` to disable detection of cloud
# provider metadata such as resource identifiers.
# discover_cloud_resource: true
```

Reduce container startup time

When you start your instrumented application, Contrast applies source transformations to both your application and the dependency code your application loads. This code rewriting increases startup time when running applications with the agent.

Starting with version 4.X, the Node.js agent includes a command line utility you can use to pre-compile applications before starting them. When started with Contrast, the pre-compiled application loads the

rewritten files from disk and significantly improves startup time. Starting with version 5, its utilities do not support command line arguments like `--agent.logger.path /dev/null`.

Use the rewriter

- In the Node.js agent's configuration file (`contrast_security.yaml`):
 - (Optional) Enable rewrite caching and specify the path of the cache location.
 - By default, the cache is enabled and will be written to the `.contrast` folder under the application directory.
 - If either `agent.node.rewrite.enable` or `agent.node.rewrite.cache.enable` are false, the CLI will throw an error.
 - (Optional) Specify the logging level.
 - The rewriter provides minimal logging at the default INFO level to inform the user which mode it is running under. At the DEBUG level, it will inform the user about failures in the rewriting process. At the TRACE level, it will log information about each file that is rewritten.
 - Enable `assess` or `protect` to specify the mode under which to run the rewriter. If both modes are enabled then the rewriter is in `assess` mode, since `assess` includes all of the transforms from `protect`.



NOTE

You can also specify modes at runtime with the `-a, --assess` or `-p, --protect` flags.

For example:

```
// example config
agent:
  logger:
    level: TRACE // Default: INFO
  node:
    rewrite:
      enable: true // default: true
      cache:
        enable: true // default: true
        path: ./path/relative/to/app/root // default: .contrast/
  assess:
    enable: true
```

- (Optional) Install the `@contrast/cli` package:
 - This can be a `devDependency` since it is not used by the agent at runtime.
 - `npm` will ask you to install the package when performing step 3 if you do not install it beforehand. This is fine, but installing it manually (and including `@contrast/cli` as a dependency in your `package.json`) will ensure you are using the specified version.

```
npm install --save-dev @contrast/cli
```

- Invoke the executable and provide it with your application's entry point (for example `index.js`).

```
npm -p @contrast/cli rewrite index.js
# or
npm -p @contrast/cli rewrite -a index.js # force assess mode
npm -p @contrast/cli rewrite -p index.js # force protect mode
```

- You can verify that the precompilation has occurred by checking the cache directory (`.contrast` by default). You should see several nested folders like `.contrast/PACKAGE_NAME/CONTRAST_REWRITER_VERSION_NUMBER/MODE`, for example, `.contrast/my-package/`

1.7.0/assess. The contents of the `_` directory in this path should match the structure of your application directory.

Known Limitations

A few scenarios cannot currently be handled by the rewriter CLI.

The rewriter CLI uses static analysis to determine the files to rewrite, so it cannot handle dynamic imports or requires. If a file is included in any of the following ways, the agent will instead rewrite that file when it is included at runtime.

```
// dynamic imports or requires
const name = 'foo.js';
await import(name);require(name);

// direct calls to createRequire
createRequire(import.meta.url)('foo.js');
// this should work, however:
const require = createRequire(import.meta.url);
require('foo.js');

// renamed require functions
const r = createRequire('...');
r('foo.js');
const myRequire = require;
myRequire('bar.js');
```

Use the Node.js agent with ESM

The Contrast Node.js agent provides full support for using ECMAScript modules (ESM) in Node.js server-side applications. ESM is the [official standard format](#) to package JavaScript client-side code.

There are two ways to run the Contrast Node.js agent if you are using ESM server-side applications.

- To explicitly assert that the code you're running is ESM and should be run as such, use the MJS file extension. Learn more about [determining a module system with file extensions](#) in the Node.js documentation.

When you use an MJS extension, Node.js knows that you've written ESM and will parse your JavaScript as such. The same is true for CJS; Node.js knows that a CJS file extension should run as CommonJS, and will parse your JavaScript as CommonJS.

- Otherwise, you can get your Node.js applications to run as ESM rather than CommonJS by including `"type": "module"` in your `package.json`, like this:

```
"main": "index.js",
"type": "module",
```

This specifically tells Node.js to parse your JS files under this `package.json` as ESM. Otherwise, by default (or when you use `"type": "commonjs"`), Node.js will parse your JS files as CommonJS.

When instrumenting an application that uses ESM (or a combination of both ESM and CJS), the way to start the application with the agent depends on the Node.js engine version and if your application is using ESM code or modules.

- **For Node LTS versions greater than or equal to 18.19.0**

Use `--import` to start the application:

```
node --import @contrast/agent app-main.mjs [app arguments]
```

- **For Node LTS versions greater than or equal to 16.17.0 and less than 18.19.0.**

Use `--loader` to start the application:

```
node --loader @contrast/agent app-main.mjs [app arguments]
```

- **For Node LTS versions less than 16.17.0. This is also applicable to applications not using the ESM syntax.**

Use the legacy method for starting the application:

```
node -r @contrast/agent app-main.js [app arguments]
```

Transpilers, compilers, source maps and the Node.js agent

The Node.js agent supports applications written in languages that compile JavaScript, such as TypeScript. Although the Node.js agent only instruments JavaScript, you can also use TypeScript if you configure the transpiler to compile your application into JavaScript.



NOTE

The source may not correspond directly with the resulting JavaScript. As a result, reported metadata (like `vulnerability line-of-code` and `filename`) references the compiled result, not the source.

Some languages, like TypeScript, require you to pre-compile your code before runtime. In these cases, the Node.js agent must point to the compiled entry point for your application.

To do this, set up the Node.js agent using the `--import` option:

```
scripts: {
  "test": "...",
  "start": "...",
  "contrast": "node --import @contrast/agent /path/to/transpiled/
entrypoint.js"
}
```

Source maps

With a source map, you can see the corresponding line numbers between the TypeScript source and the transpiled JavaScript.

To use source maps, you must enable the Babel re-writer and rewrite caching in your YAML configuration:

```
agent:
  node:
    rewrite_cache:
      enable: true
      path: ./cache
    enable_babel: true
```

When enabled, the agent looks for source maps (MAP files) in the same directory as the source that's being loaded (for example, if the file `/home/app/index.js` is loaded, then Contrast looks for `/home/app/index.js.map`).

If you do not already have source maps then it needs to be recompiled using the necessary flags to produce source maps. This is different for every transpiler, so check the options for your transpiler. For example, for TypeScript, append the `--source-map` flag to the TypeScript compiler (`tsc`) or add the `"sourceMap": true` entry to the `"compilerOptions"` section in `tsconfig.json`.

Node.js telemetry

The Node.js agent uses telemetry to collect usage data. Telemetry is collected when an instrumented application first loads the agent's sensors. Currently, data is only sent at startup. In the future, the agent will be able to send errors and metrics when the instrumented app is running and being exercised.

[Your privacy is important to us \(page 1275\)](#). The telemetry feature does not collect application data. The data is anonymized before being sent securely to Contrast. Then the aggregated data is stored encrypted and under restricted access control. Any collected data will be deleted after one year.

The telemetry feature collects the following data:

Agent versions	Data
@contrast/agent 4.12.0 and later	Agent version
	Operating system and version
	Node.js version
	Is the app running in a container (Y/N)

To opt-out of the telemetry feature, set the `CONTRAST_AGENT_TELEMETRY_OPTOUT` environment variable to `1` or `true`.

Telemetry data is securely sent to <http://telemetry.nodejs.contrastsecurity.com>. You can also opt-out of telemetry by blocking communication at the network level.

PHP agent

The Contrast PHP agent analyzes PHP web applications at runtime for library usage and vulnerability detection. The PHP agent is implemented as a PHP extension.



NOTE

The PHP agent supports Assess, Protect, and SCA.

As a next step, you can:

- [Install the PHP agent \(page 392\)](#)
- [View PHP agent supported technologies \(page 391\)](#)
- [View PHP agent system requirements \(page 392\)](#)

PHP agent supported technologies

We support the following technologies for this agent.

Technology	Supported versions	Notes
Language version	• 7.4, 8.0, 8.1, 8.2, 8.3	The agent depends on the <code>mbstring</code> and <code>curl</code> extensions. Support only for NTS versions of PHP.
Application frameworks	• Laravel • Symfony • Drupal 8, 9, 10, 11	

Technology	Supported versions	Notes
Servers	<ul style="list-style-type: none">• Apache	<p>Other servers using <code>mod_php</code> or <code>php-fpm</code> may work but are not currently supported.</p> <p>If you are using <code>php-fpm</code>, set <code>clear_env</code> to <code>no</code>. This setting ensures that the required environment variables are available to the PHP worker processes. If you use the default value, all shell environment variables are cleared before they reach the PHP worker processes, which prevents the agent from functioning correctly.</p> <p>You can usually find the <code>php-fpm</code> pool configuration file in <code>/usr/local/etc/php-fpm.d/www.conf</code>.</p>
Package manager	<ul style="list-style-type: none">• Composer	Supported by SCA analysis.
Content management system	<ul style="list-style-type: none">• WordPress 6	

PHP agent system requirements

Before installing the PHP agent, your system must meet the following requirements:

Requirement	Version
Runtime system	<ul style="list-style-type: none">• 64-bit Linux
Operating systems	<ul style="list-style-type: none">• CentOS 7, 8• RHEL 7, 8, 9• Debian (Ubuntu) 18.4 and higher
Processor architecture	<ul style="list-style-type: none">• AMD64

Install the PHP agent

A basic installation of the PHP agent looks like this:

1. Install the agent package.
2. Download the `contrast_security.yaml` and place it in the proper path.
3. Configure the PHP interpreter to enable the Contrast agent extension.
4. Exercise and test your application.
5. Verify that Contrast sees your application.

For specific installation instructions, select one of the following options:

- [Install PHP with Debian \(page 392\)](#)
- [Install PHP with RPM \(page 394\)](#)
- [Install PHP with Lando \(page 396\)](#)

Install PHP agent with Debian

Steps

To install the PHP agent:

1. Install the agent package from <https://pkg.contrastsecurity.com>. This command registers our package repository in your system:

```
curl \
  https://pkg.contrastsecurity.com/api/gpg/key/public | sudo apt-key add -

echo "deb https://pkg.contrastsecurity.com/debian-public/ $(sed -rne 's/^VERSION_CODENAME=(.*)$/\1/p' /etc/*ease) contrast" \
  | sudo tee /etc/apt/sources.list.d/contrast.list
```

```
echo "deb https://pkg.contrastsecurity.com/debian-public/ all contrast" \
| sudo tee -a /etc/apt/sources.list.d/contrast.list
```

2. Once complete, use this command to install the agent. You may need to use the `sudo` command depending on the permissions in your environment:

```
apt-get update && apt-get install contrast-php-agent
```

3. Continue with configuration.

To configure the PHP agent:

There are two ways to configure the agent to work in your environment.

1. Use the `contrast-php-util` command. This command installs along with the PHP agent and can be used from the command line to enable the agent in your environment.

```
contrast-php-util enable-agent
```

This will create an `ini` file in the scan directory used by PHP to manage extensions.

OR

2. Edit the PHP configuration file, called `php.ini`. It can be found on many systems under `/usr/local/etc/php/`.

If the `php-config` command is available, it can be used to find the configuration file path using `php-config --ini-path`. If no configuration file yet exists under that path, it will need to be created.

```
echo "extension=/usr/local/lib/contrast/php/contrast.so" >> php-config --
ini-path/php.ini
```

On systems in which the `php-config` command is not available PHP itself can be used to tell you where the `ini` file is located:

```
php -i | grep php.ini
```

The Configuration File (`php.ini`) Path line will tell you which `ini` file is active. You can then edit that file to add this to the end of the file:

```
extension=/usr/local/lib/contrast/php/contrast.so
```

Note that in certain environments, the `ini` file used by the command line instance of PHP might be different from that loaded in the server environment. FPM will use an `ini` file often located in the `/etc/php/<php version>/fpm/` directory. Please ensure that you are operating on the proper PHP instance.

3. Continue with the final setup.

To finalize setup:

1. Ensure that the necessary extensions are enabled.
The PHP agent requires that the `mbstring`, `json`, and `curl` PHP extensions be installed and enabled. If they are not, edit the `php.ini` config file as above and ensure that these lines are present in the configuration file:

```
extension=curl
extension=mbstring
extension=json
```

2. [Configure the PHP agent \(page 397\)](#) using the [PHP YAML template \(page 397\)](#) or environment variables (if not already done).

3. Start your application in the normal way.
4. Exercise and test your application.
5. Verify that the PHP agent is running by checking the Contrast web interface and/or looking for the PHP agent log output (depending on configuration).

Notes

- Library analysis and route discovery are currently performed on the first request to the application. For this reason, we expect the first request to be considerably slower than subsequent requests.
- The agent has not been tested with third-party PHP extensions. The behavior of the agent with any other third-party extensions (including APMs, etc.) is undefined. The agent is known to be incompatible with xdebug and should not be used concurrently.
- The agent may not work properly when preloading is enabled. Disabling preloading when using the agent is recommended.
- By default, the agent assumes that the server's working directory when it runs the PHP application is the same as the top-level directory of the application source tree. The agent uses this path to perform library analysis and route discovery. If this is not the case, you will need to use the `application.path` setting in the configuration to set the top-level working directory of your application.

Install PHP agent with Red Hat Package Manager (RPM)

Steps

To install the PHP agent:

1. Install the agent package from <https://pkg.contrastsecurity.com>. Use this script in your shell to configure your RPM-based system for our package repository. You may need `sudo` permissions.

```
tee /etc/yum.repos.d/contrast.repo <<-"EOF"
[contrast]
name=Contrast centos-$releasever repo
baseurl=https://pkg.contrastsecurity.com/rpm-public/redhat-
$releasever/
gpgcheck=0
enabled=1
EOF
```

2. Once complete, use one of these commands to install the agent and service. The command used depends on which platform you are installing the agent:

```
sudo yum install contrast-php-agent
```

OR

```
sudo dnf install contrast-php-agent
```

3. Continue with configuration.

To configure the PHP agent:

There are two ways to configure the agent to work in your environment.

1. Use the `contrast-php-util` command. This command installs along with the PHP agent and can be used from the command line to enable the agent in your environment.

```
contrast-php-util enable-agent
```

This will create an `ini` file in the scan directory used by PHP to manage extensions

OR

2. Edit the PHP configuration file, called `php.ini`. On many systems it can be found under `/usr/local/etc/php/`.

If the `php-config` command is available, it can be used to find the configuration file path using `php-config --ini-path`. If no configuration file yet exists under that path, it will need to be created.

```
echo "extension=/usr/local/lib/contrast/php/contrast.so" >> php-config --ini-path/php.ini
```

On systems in which the `php-config` command is not available PHP itself can be used to tell you where the `ini` file is located:

```
php -i | grep php.ini
```

The Configuration File (`php.ini`) Path line will tell you which `ini` file is active. You can then edit that file to add this to the end of the file:

```
extension=/usr/local/lib/contrast/php/contrast.so
```

Note that in certain environments, the `ini` file used by the command line instance of PHP might be different from that loaded in the server environment. FPM will use an `ini` file often located in the `/etc/php/<php version>/fpm/` directory. Please ensure that you are operating on the proper PHP instance.

3. Continue with the final setup.

To finalize setup:

1. Ensure that the necessary extensions are enabled.

The PHP agent requires that the `mbstring`, `json`, and `curl` PHP extensions be installed and enabled. If they are not, edit the `php.ini` config file as above and ensure that these lines are present in the configuration file:

```
extension=curl
extension=mbstring
extension=json
```

2. [Configure the PHP agent \(page 397\)](#) using the [PHP YAML template \(page 397\)](#) or environment variables (if not already done).
3. Start your application in the normal way.
4. Exercise and test your application.
5. Verify that the PHP agent is running by checking the Contrast web interface and/or looking for the PHP agent log output (depending on configuration).

Notes

- Library analysis and route discovery are currently performed on the first request to the application. For this reason, we expect the first request to be considerably slower than subsequent requests.
- The agent has not been tested with third-party PHP extensions. The behavior of the agent with any other third-party extensions (including APMs, etc.) is undefined. The agent is known to be incompatible with `xdebug` and should not be used concurrently.
- The agent may not work properly when preloading is enabled. Disabling preloading when using the agent is recommended.
- By default, the agent assumes that the server's working directory when it runs the PHP application is the same as the top-level directory of the application source tree. The agent uses this path to perform library analysis and route discovery. If this is not the case, you will need to use the `application.path` setting in the configuration to set the top-level working directory of your application.

Install PHP agent on a Lando application server

Before you begin

- [Install Lando](#).
- Create a working application and `.lando.yml` file using [supported technologies \(page 391\)](#).

Steps



IMPORTANT

Never place the `contrast.env` file in the application's webroot directory.

1. In your project's root directory, under the `appserver` service section of the `.lando.yml` file, install the Contrast agent in the default service by adding this configuration:

```
services:
  appserver:
    build_as_root:
      - curl https://pkg.contrastsecurity.com/api/gpg/key/public | apt-
key add -
      - echo "deb https://pkg.contrastsecurity.com/debian-public/ $
(sed -rne 's/^VERSION_CODENAME=(.*)$/\1/p' /etc/*ease) contrast" \ |
tee /etc/apt/sources.list.d/contrast.list
      - echo "deb https://pkg.contrastsecurity.com/debian-public/ all
contrast" | tee -a /etc/apt/sources.list.d/contrast.list
      - apt-get update && apt-get install contrast-php-agent
      - contrast-php-util enable-agent
env_file:
  - /path/to/contrast.env
```

2. The environment variables are set in the `contrast.env` file for either an agent token or a legacy configuration. The following variables you can set are proposed changes for your reference.

- **Agent Token:** In a `contrast.env` file outside the webroot of your project, enable the connection to Contrast by adding the minimum agent configuration:

```
CONTRAST__API__TOKEN={token value}
CONTRAST__ASSESS__ENABLE=true
CONTRAST__SERVER__NAME={yourServerName}
CONTRAST__AGENT__LOGGER__PATH=stdout
```

If your agent configuration refers to both the legacy settings and the agent token, (in environment variables or the YAML file), the legacy settings take precedence. Remove references to the legacy settings to use just the agent token value.

- **Legacy settings:** If you are using a PHP agent version earlier than 1.34.0, set these variables:

```
CONTRAST__API__URL={contrastURL}
CONTRAST__API__API_KEY={apiKey}
CONTRAST__API__SERVICE_KEY={serviceKey}
CONTRAST__API__USER__NAME={contrastAgentUserName}
CONTRAST__ASSESS__ENABLE=true
CONTRAST__SERVER__NAME={yourServerName}
CONTRAST__AGENT__LOGGER__PATH=stdout
```




NOTES

- Configuring a name for the `server.name` property prevents the agent from creating a new server entry in the Contrast web interface when the server restarts.
- The `agent.logger.path` property is set to `stdout` to avoid printing agent logs to the `webroot` directory. The Contrast agent logs are available using this configuration by running the `lando logs -s appserver` command.
- You can use the [Contrast agent configuration editor \(page 109\)](#) to verify and export the agent configuration as environment variables.

3. To enable the Contrast agent, run the `lando start` command (if not used previously) or the `lando rebuild -y` command.

Configure the PHP agent

The [standard configuration \(page 102\)](#) for all agents uses this [order of precedence \(page 106\)](#).

You may use a YAML configuration file or environment variables to configure the agent when running your application:

- You can use [environment variables \(page 110\)](#) to configure your build
- You can create your own YAML configuration file or use this [YAML template \(page 397\)](#) that contains all valid properties for the PHP agent



TIP

Use the [Contrast agent configuration editor \(page 109\)](#) to create or upload a YAML configuration file, validate YAML and get setting recommendations.

PHP YAML template

Use this template to configure the PHP agent using a YAML configuration file. (Learn more about [YAML configuration \(page 107\)](#).)

Place your YAML file in the default location: `/etc/contrast/php/contrast_security.yaml`
or `/etc/contrast/contrast_security.yaml`.

```
#
=====
==
# Use the properties in this YAML file to configure a Contrast agent.
# Go to https://docs.contrastsecurity.com/en/order-of-precedence.html
# to determine the order of precedence for configuration values.
#
=====
==

# Use this setting if you want to temporarily disable a Contrast agent.
# Set to `true` to enable the agent; set to `false` to disable the agent.
# enable: true
```

```
#
=====
==
# api
# Use the properties in this section to connect the agent to the Contrast
# UI.
#
=====
==
api:

# ***** REQUIRED *****
# Set the URL for the Contrast UI.
url: https://app.contrastsecurity.com/Contrast

# ***** REQUIRED *****
# Set the API key needed to communicate with the Contrast UI.
api_key: NEEDS_TO_BE_SET

# ***** REQUIRED *****
# Set the service key needed to communicate with the Contrast
# UI. It is used to calculate the Authorization header.
service_key: NEEDS_TO_BE_SET

# ***** REQUIRED *****
# Set the user name used to communicate with the Contrast
# UI. It is used to calculate the Authorization header.
user_name: NEEDS_TO_BE_SET

# base64 encoded JSON object containing the `url`,
# `api_key`, `service_key`, and `user_name` config options,
# allowing them all to be set in a single variable.
# token: NEEDS_TO_BE_SET

#
=====
# api.certificate
# Use the following properties for communication
# with the Contrast UI using certificates.
#
=====
# certificate:

# If set to `false`, the agent will ignore the
# certificate configuration in this section.
# enable: true

# Set the absolute or relative path to the Certificate
# PEM file for communication with the Contrast UI.
# cert_file: NEEDS_TO_BE_SET

#
=====
# api.proxy
# Use the following properties for communication
```

```
# with the Contrast UI over a proxy.
#
=====
# proxy:

# Set value to `true` for the agent to communicate
# with the Contrast web interface over a proxy. Set
# value to `false` if you don't want to use the proxy.
# enable: NEEDS_TO_BE_SET

# Set the URL for your Proxy Server. The URL form is `scheme://
host:port`,
# or scheme://username:password@host:port, if you need to
# specify a username and password to connect through the proxy.
# url: NEEDS_TO_BE_SET

#
=====
==
# agent
# Use the properties in this section to control the way and frequency
# with which the agent communicates to logs and the Contrast UI.
#
=====
==
# agent:

#
=====
# agent.diagnostics
# Use the properties in this section to specify the information the agent
# should collect and report in order to diagnose problems in the agent.
#
#
=====
# diagnostics:

# Creates config and system info files
# at startup if true. True by default.
#
# The same thing can be achieved by setting the
# CONTRAST__AGENT__DIAGNOSTICS__ENABLE=[true/false] env variable.
#
# enable: true

# Set the directory in which to write diagnostic files.
# Defaults to the application's current working directory.
# report_path: ./

#
=====
# agent.logger
# Define the following properties to set logging values.
# If the following properties are not defined, the
# agent uses the logging values from the Contrast UI.
```

```
#
=====
# logger:

# Enable diagnostic logging by setting a path to a log file.
# While diagnostic logging hurts performance, it generates
# useful information for debugging Contrast. The value set here
# is the location to which the agent saves log output. If no
# log file exists at this location, the agent creates a file.
#
# Example - `/opt/Contrast/contrast.log` creates a log in the
# `/opt/Contrast` directory, and rotates it automatically as needed.
#
# path: ./contrast_agent.log

# Set the the log output level. Valid options are
# `ERROR`, `WARN`, `INFO`, `DEBUG`, and `TRACE`.
# level: INFO

# Override the name of the process the agents uses in logs.
# progname: Contrast Agent

# Set the roll size for log files in megabytes. The agent will
# attempt to prevent the log file from being larger than this size.
# roll_size: 100

#
=====
# agent.php
# The following properties apply to any PHP agent-wide configurations.
#
=====
# php:

# Specify the path to the PHP executable you want to use.
# cli_executable: NEEDS_TO_BE_SET

#
=====
==
# inventory
# Use the properties in this section to override the inventory features.
#
=====
==
# inventory:

# Set to `false` to disable inventory features in the agent.
# enable: true

# Set to `false` to disable library analysis.
# analyze_libraries: true

# Apply a list of labels to libraries. Labels
# must be formatted as a comma-delimited list.
```

```
# Example - `label1, label2, label3`
#
# tags: NEEDS_TO_BE_SET

#
=====
==
# assess
# Use the properties in this section to control Assess.
#
=====
==
# assess:

# Include this property to determine if the Assess
# feature should be enabled. If this property is not
# present, the decision is delegated to the Contrast UI.
# enable: false

# Apply a list of labels to vulnerabilities and preflight
# messages. Labels must be formatted as a comma-delimited list.
# Example - `label1, label2, label3`
#
# tags: NEEDS_TO_BE_SET

# Value options are `ALL`, `SOME`, or `NONE`.
# stacktraces: ALL

#
=====
# assess.rules
# Use the following properties to control simple rule configurations.
#
=====
# rules:

# Define a list of Assess rules to disable in the agent. To view a
# list of rule names, in Contrast go to user menu > Policy Management >
# Assess rules. The rules must be formatted as a comma-delimited list.
#
# Example - Set `reflected-xss,sql-injection` to disable
# the reflected-xss rule and the sql-injection rule.
#
# disabled_rules: NEEDS_TO_BE_SET

#
=====
==
# application
# Use the properties in this section for
# the application(s) hosting this agent.
#
=====
==
# application:
```

```
# Override the reported application name.
#
# Note - On systems where multiple, distinct applications may be served
# by a single process, this configuration causes the agent to report
# all discovered applications as one application with the given name.
#
# Note - On Java systems where multiple, distinct applications may be
# served by a single process, this configuration causes the agent to
report
# all discovered applications as one application with the given name.
#
# name: NEEDS_TO_BE_SET

# Override the reported application path.
# path: NEEDS_TO_BE_SET

# Add the name of the application group with which this
# application should be associated in the Contrast UI.
# group: NEEDS_TO_BE_SET

# Add the application code this application should use in the Contrast UI.
# code: NEEDS_TO_BE_SET

# Override the reported application version.
# version: NEEDS_TO_BE_SET

# Apply labels to an application. Labels must
# be formatted as a comma-delimited list.
# Example - `label1,label2,label3`
#
# tags: NEEDS_TO_BE_SET

# Define a set of `key=value` pairs (which conforms to RFC 2253) for
# specifying user-defined metadata associated with the application. The
# set must be formatted as a comma-delimited list of `key=value` pairs.
# Example - `business-unit=accounting, office=Baltimore`
#
# metadata: NEEDS_TO_BE_SET

# Provide the ID of a session which already exists in the Contrast
# UI. Vulnerabilities discovered by the agent are associated with
# this session. If an invalid ID is supplied, the agent will be
# disabled. This option and `application.session_metadata` are
# mutually exclusive; if both are set, the agent will be disabled.
# session_id: NEEDS_TO_BE_SET

# Provide metadata which is used to create a new session ID in the
# Contrast UI. Vulnerabilities discovered by the agent are associated with
# this new session. This value should be formatted as `key=value` pairs
# (conforming to RFC 2253). Available key names for this configuration
# are branchName, buildNumber, commitHash, committer, gitTag, repository,
# testRun, and version. This option and `application.session_id` are
# mutually exclusive; if both are set the agent will be disabled.
# session_metadata: NEEDS_TO_BE_SET
```

```
#
=====
==
# server
# Use the settings in this section to set metadata for the server
# hosting this agent. Contrast recognizes common, supported server
# names, paths, types and environments. Doing this may require a new
# server or license, and it may affect functionality of some features.
#
=====
==
# server:

# Override the reported server name.
# name: localhost

# Override the reported server path.
# path: NEEDS_TO_BE_SET

# Override the reported server type.
# type: NEEDS_TO_BE_SET

# Set the environment directly to override the default set
# by the Contrast UI. This allows the user to configure the
# environment dynamically at startup rather than manually
# updating the Server in the Contrast UI themselves afterwards.
#
# Valid values include `QA`, `PRODUCTION` and `DEVELOPMENT`.
# For example, `PRODUCTION` registers this Server as
# running in a `PRODUCTION` environment, regardless of the
# organization's default environment in the Contrast UI.
#
# environment: NEEDS_TO_BE_SET

# Apply a list of labels to the server. Labels
# must be formatted as a comma-delimited list.
# Example - `label1,label2,label3`
#
# tags: NEEDS_TO_BE_SET

# Set to `false` to disable detection of cloud
# provider metadata such as resource identifiers.
# discover_cloud_resource: true
```

```
#
=====
==
# Use the properties in this YAML file to configure a Contrast agent.
# Go to https://docs.contrastsecurity.com/en/order-of-precedence.html
# to determine the order of precedence for configuration values.
#
=====
==
```

```
# Use this setting if you want to temporarily disable a Contrast agent.
# Set to `true` to enable the agent; set to `false` to disable the agent.
# enable: true

#
=====
==
# api
# Use the properties in this section to connect the agent to the Contrast
UI.
#
=====
==
api:

# ***** REQUIRED *****
# Set the URL for the Contrast UI.
url: https://app.contrastsecurity.com/Contrast

# ***** REQUIRED *****
# Set the API key needed to communicate with the Contrast UI.
api_key: NEEDS_TO_BE_SET

# ***** REQUIRED *****
# Set the service key needed to communicate with the Contrast
# UI. It is used to calculate the Authorization header.
service_key: NEEDS_TO_BE_SET

# ***** REQUIRED *****
# Set the user name used to communicate with the Contrast
# UI. It is used to calculate the Authorization header.
user_name: NEEDS_TO_BE_SET

# base64 encoded JSON object containing the `url`,
# `api_key`, `service_key`, and `user_name` config options,
# allowing them all to be set in a single variable.
# token: NEEDS_TO_BE_SET

#
=====
# api.certificate
# Use the following properties for communication
# with the Contrast UI using certificates.
#
=====
# certificate:

# If set to `false`, the agent will ignore the
# certificate configuration in this section.
# enable: true

# Set the absolute or relative path to the Certificate
# PEM file for communication with the Contrast UI.
# cert_file: NEEDS_TO_BE_SET
```



```
#
=====
# api.proxy
# Use the following properties for communication
# with the Contrast UI over a proxy.
#
=====
# proxy:

# Set value to `true` for the agent to communicate
# with the Contrast web interface over a proxy. Set
# value to `false` if you don't want to use the proxy.
# enable: NEEDS_TO_BE_SET

# Set the URL for your Proxy Server. The URL form is `scheme://
host:port`,
# or scheme://username:password@host:port, if you need to
# specify a username and password to connect through the proxy.
# url: NEEDS_TO_BE_SET

#
=====
==
# agent
# Use the properties in this section to control the way and frequency
# with which the agent communicates to logs and the Contrast UI.
#
=====
==
# agent:

#
=====
# agent.diagnostics
# Use the properties in this section to specify the information the agent
# should collect and report in order to diagnose problems in the agent.
#
#
=====
# diagnostics:

# Creates config and system info files
# at startup if true. True by default.
#
# The same thing can be achieved by setting the
# CONTRAST__AGENT__DIAGNOSTICS__ENABLE=[true/false] env variable.
#
# enable: true

# Set the directory in which to write diagnostic files.
# Defaults to the application's current working directory.
# report_path: ./

#
```

```
=====
# agent.logger
# Define the following properties to set logging values.
# If the following properties are not defined, the
# agent uses the logging values from the Contrast UI.
#
=====
# logger:

# Enable diagnostic logging by setting a path to a log file.
# While diagnostic logging hurts performance, it generates
# useful information for debugging Contrast. The value set here
# is the location to which the agent saves log output. If no
# log file exists at this location, the agent creates a file.
#
# Example - `/opt/Contrast/contrast.log` creates a log in the
# `/opt/Contrast` directory, and rotates it automatically as needed.
#
# path: ./contrast_agent.log

# Set the the log output level. Valid options are
# `ERROR`, `WARN`, `INFO`, `DEBUG`, and `TRACE`.
# level: INFO

# Override the name of the process the agents uses in logs.
# progname: Contrast Agent

# Set the roll size for log files in megabytes. The agent will
# attempt to prevent the log file from being larger than this size.
# roll_size: 100

#
=====
# agent.php
# The following properties apply to any PHP agent-wide configurations.
#
=====
# php:

# Specify the path to the PHP executable you want to use.
# cli_executable: NEEDS_TO_BE_SET

#
=====
==
# inventory
# Use the properties in this section to override the inventory features.
#
=====
==
# inventory:

# Set to `false` to disable inventory features in the agent.
# enable: true
```

```
# Set to `false` to disable library analysis.
# analyze_libraries: true

# Apply a list of labels to libraries. Labels
# must be formatted as a comma-delimited list.
# Example - `label1, label2, label3`
#
# tags: NEEDS_TO_BE_SET

#
=====
==
# assess
# Use the properties in this section to control Assess.
#
=====
==
# assess:

# Include this property to determine if the Assess
# feature should be enabled. If this property is not
# present, the decision is delegated to the Contrast UI.
# enable: false

# Apply a list of labels to vulnerabilities and preflight
# messages. Labels must be formatted as a comma-delimited list.
# Example - `label1, label2, label3`
#
# tags: NEEDS_TO_BE_SET

# Value options are `ALL`, `SOME`, or `NONE`.
# stacktraces: ALL

#
=====
# assess.rules
# Use the following properties to control simple rule configurations.
#
=====
# rules:

# Define a list of Assess rules to disable in the agent. To view a
# list of rule names, in Contrast go to user menu > Policy Management >
# Assess rules. The rules must be formatted as a comma-delimited list.
#
# Example - Set `reflected-xss,sql-injection` to disable
# the reflected-xss rule and the sql-injection rule.
#
# disabled_rules: NEEDS_TO_BE_SET

#
=====
==
# application
# Use the properties in this section for
```

```
# the application(s) hosting this agent.
#
=====
==
# application:

# Override the reported application name.
#
# Note - On systems where multiple, distinct applications may be served
# by a single process, this configuration causes the agent to report
# all discovered applications as one application with the given name.
#
# Note - On Java systems where multiple, distinct applications may be
# served by a single process, this configuration causes the agent to
report
# all discovered applications as one application with the given name.
#
# name: NEEDS_TO_BE_SET

# Override the reported application path.
# path: NEEDS_TO_BE_SET

# Add the name of the application group with which this
# application should be associated in the Contrast UI.
# group: NEEDS_TO_BE_SET

# Add the application code this application should use in the Contrast UI.
# code: NEEDS_TO_BE_SET

# Override the reported application version.
# version: NEEDS_TO_BE_SET

# Apply labels to an application. Labels must
# be formatted as a comma-delimited list.
# Example - `label1,label2,label3`
#
# tags: NEEDS_TO_BE_SET

# Define a set of `key=value` pairs (which conforms to RFC 2253) for
# specifying user-defined metadata associated with the application. The
# set must be formatted as a comma-delimited list of `key=value` pairs.
# Example - `business-unit=accounting, office=Baltimore`
#
# metadata: NEEDS_TO_BE_SET

# Provide the ID of a session which already exists in the Contrast
# UI. Vulnerabilities discovered by the agent are associated with
# this session. If an invalid ID is supplied, the agent will be
# disabled. This option and `application.session_metadata` are
# mutually exclusive; if both are set, the agent will be disabled.
# session_id: NEEDS_TO_BE_SET

# Provide metadata which is used to create a new session ID in the
# Contrast UI. Vulnerabilities discovered by the agent are associated with
# this new session. This value should be formatted as `key=value` pairs
```

```
# (conforming to RFC 2253). Available key names for this configuration
# are branchName, buildNumber, commitHash, committer, gitTag, repository,
# testRun, and version. This option and `application.session_id` are
# mutually exclusive; if both are set the agent will be disabled.
# session_metadata: NEEDS_TO_BE_SET

#
=====
==
# server
# Use the settings in this section to set metadata for the server
# hosting this agent. Contrast recognizes common, supported server
# names, paths, types and environments. Doing this may require a new
# server or license, and it may affect functionality of some features.
#
=====
==
# server:

# Override the reported server name.
# name: localhost

# Override the reported server path.
# path: NEEDS_TO_BE_SET

# Override the reported server type.
# type: NEEDS_TO_BE_SET

# Set the environment directly to override the default set
# by the Contrast UI. This allows the user to configure the
# environment dynamically at startup rather than manually
# updating the Server in the Contrast UI themselves afterwards.
#
# Valid values include `QA`, `PRODUCTION` and `DEVELOPMENT`.
# For example, `PRODUCTION` registers this Server as
# running in a `PRODUCTION` environment, regardless of the
# organization's default environment in the Contrast UI.
#
# environment: NEEDS_TO_BE_SET

# Apply a list of labels to the server. Labels
# must be formatted as a comma-delimited list.
# Example - `label1,label2,label3`
#
# tags: NEEDS_TO_BE_SET

# Set to `false` to disable detection of cloud
# provider metadata such as resource identifiers.
# discover_cloud_resource: true
```

Python agent

The Python agent enables interactive application security testing (IAST) and runtime application self-protection (RASP) for Python applications. It provides support for the most popular Python

web application frameworks and also strives to be compatible with any other Python application or framework that conforms to either the [WSGI](#) or [ASGI](#) standards.

In Assess (IAST), the agent identifies vulnerable dataflow paths and other issues during the normal execution of your application. It reports these findings to your organization in Contrast where you can remediate the vulnerabilities before deploying the application in a live environment.

In Protect (RASP), the Python agent inspects HTTP requests to identify potentially harmful input vectors. During the request, the agent inspects database queries, file writes, and other potentially damaging actions resulting from the request. At the end of the request, the agent inspects the rendered output for successful attacks and can block a successful attack from being forwarded to the application user. Instead, the agent natively processes all the analysis internally, reaping many benefits including performance gains.



NOTE

The Python agent supports Assess, Protect, and SCA.

As a next step, you can:

- [Install the agent \(page 411\)](#)
- [View supported technologies for the Python agent \(page 410\)](#)

Supported technologies for the Python agent

We support the following technologies for this agent.

Technology	Supported versions	Notes
Language version	<ul style="list-style-type: none">• 3.13.X: First supported agent was 9.8.0• 3.12.X: First supported agent was 5.27.0• 3.11.X: First supported agent was 5.19.0• 3.10.X: First supported agent was 5.2.0• 3.9.X: First supported agent was 4.2.0	<p>Contrast supports Python Long-Term Support (LTS) versions in bugfix and security status. Support for Python versions is shifted as the working group shifts its LTS window.</p> <p>Not supported:</p> <ul style="list-style-type: none">• 3.6.X, 3.5X, and 2.7.X: Last supported agent was 4.14.3• 3.7.X: Last supported agent was 5.27.0• 3.8.X: Last supported agent was 9.8.0• Contrast does not currently support Python's experimental JIT compiler or free-threading. The agent will not run in an environment where either of these features is enabled.

Technology	Supported versions	Notes
Application frameworks	<ul style="list-style-type: none">• Aiohttp 3.7 - 3.9• Bottle: 0.13• Django: 3.2 - 5.1• Django Rest Framework: 3.12 - 3.15• Falcon: 3.0 - 3.1• FastAPI: 0.71 - 0.111• Flask: 1.1 - 3.1• Pyramid: 1.10 - 2.0• Quart: 0.15 - 0.20	<p>The Python agent is meant to be WSGI-compatible. It may be compatible with other WSGI applications as long as the guidelines are followed.</p> <p>The Agent is also compatible with frameworks that provide ASGI interfaces including Django, FastAPI, and Quart.</p> <p>FastAPI and Starlette (on which FastAPI depends) are relatively new libraries that are undergoing continuous development changes, which may break Contrast support. FastAPI states that development moves quickly. Contrast will maintain support up to the stated version and update documentation when new support is released.</p> <p>Micro/patch versions are omitted for simplicity, but we encourage and only explicitly test the latest patch within each minor version series. For example, 3.2 refers to the latest 3.2.x version of the package available on PyPI.</p>
Platforms	<ul style="list-style-type: none">• macOS: x86-64 / ARM64• manylinux: x86-64 / ARM64• musllinux (alpine): x86-64 / ARM64	On Alpine systems, <code>libgcc</code> is required. This can be installed with <code>apk add libgcc</code> .
Web servers	<ul style="list-style-type: none">• Gunicorn 0.16.1 –21.2.X• uWSGI 2.0.14 - 2.0.X• Uvicorn: 0.14.X - 0.23.X	
Databases	<ul style="list-style-type: none">• Mongo (pymongo)• MySQL (PyMySQL and mysql-connector)• PostgreSQL (psycopg2)• SQLite3 (sqlite3 and pysqlite2)	
Object-relational mapping databases (ORM)	<ul style="list-style-type: none">• Flask-SQLAlchemy• SQLAlchemy	

Install the Python agent

The Python agent is installed as a standard Python package.

In earlier versions (before version 5.19.0), the Python Agent uses the Contrast Service to communicate its results. By default, the service is started automatically when your application starts. Configuring the agent to communicate with a standalone [Contrast service \(page 549\)](#) that runs independently is also possible.

In version 5.19.0 and later, the Python Agent does not use the Contrast Service.

[Install the agent with PyPi \(page 411\)](#) or [update the Python agent \(page 412\)](#).

Install the Python agent with PyPI

To install the Python agent with [PyPI](#):

1. Verify that a C compiler and C standard library headers are available on the system where you are installing the agent. Package names may differ across platforms.
2. Install the agent using pip.

```
pip install contrast-agent
```

**TIP**

If you have a `requirements.txt` file, you can add `contrast-agent` to that file, and install with `pip install -r requirements.txt`.

3. [Configure the agent. \(page 413\)](#)
4. Start and exercise your application using the [Contrast Runner \(page 467\)](#).
5. Verify that your server is registered in Contrast and that it reports an instance of your application.

Python update agent

The most reliable and effective way to automatically update the Contrast Python agent is to use the Python `pip` package installer to install and download the latest version available. Because `pip` manages all dependencies for your Python application, it should already be available and part of your build environment. How frequently you update the Contrast Python agent and where you get updates depends on your organization's preferences and your Contrast implementation: hosted or on-premises.

The main steps are:

1. Choose a source for the Contrast Python agent.
2. Install the agent.
3. Use scripts for automatic updates.

Before you begin

- Access to the PyPI repository for the Contrast agent.
- Confirmed that your Python application runs properly without the Contrast Python agent.
- Previously successfully installed the Contrast Python agent.
- Defined a policy for how and when to update the agent, based on your change management policy and the environment where you deploy agents.

Install the agent and use scripts for automatic updates

1. Choose a source for the Python agent:
 - PyPI public (or private) repository
2. Specify the Contrast Python agent as a dependency in `requirements.txt`.
`requirements.txt` is the file where you specify which dependencies you want to automatically resolve every time your Python application builds with artifacts from PyPI (public or private). Include the Contrast Python agent here to easily keep every new build of your application aligned with the latest version of the agent. Do not specify a version for `contrast-agent`, and it will retrieve the latest version.
3. After you update `requirements.txt`, use the following command when you build your application. This will automatically download and add or update the Contrast Python agent from PyPI to the Python application:

```
$ pip install -U -r requirements.txt
```

Install and update manually

For some organizations, the `requirements.txt` file must be consistent across environments, or they do not plan to install the Contrast Python agent into all environments. In these cases, install the agent manually. You can manually update agents as part of a Python build process.

1. Choose a source for the Python agent:
 - PyPI public (or private) repository
2. Use the following command to manually retrieve and add or update the Contrast Python agent from PyPI (public or private) to the Python application:


```
$ pip install -U contrast-agent
```

See also

- [Python supported technologies \(page 410\)](#)
- [Install Python \(page 411\)](#)

Configure the Python agent

The [standard configuration \(page 102\)](#) for all agents uses this [order of precedence \(page 106\)](#).

As of version 5.24.0 of the agent, the [Contrast Runner \(page 467\)](#) is the recommended way to use the Python Agent for most supported frameworks. Manual [middleware configuration \(page 436\)](#) is still supported for backward compatibility and may continue to be necessary for certain frameworks and applications.

Contrast Service Configuration (before version 5.19.0 only)

In earlier versions (before version 5.19.0), the Python agent launches an executable on startup that also needs access to the configuration files. Since the service is generally launched by the Python agent process, it has access to the same configuration file as the agent. However, if the service is started independently, it will attempt to use the same [order of precedence \(page 106\)](#) for its configuration file.

In other words, the service can share the application's configuration file, if (as is usually the case) the service's working directory is also the base directory of the application. Both the agent and the service use the `/etc/contrast/contrast_security.yaml` path.



TIP

Use the [Contrast agent configuration editor \(page 109\)](#) to create or upload a YAML configuration file, validate YAML, and get setting recommendations.

Python YAML template

Use this template to configure the Python agent using a YAML configuration file. (Learn more about [YAML configuration \(page 107\)](#).)

Place your YAML file in the default location: `/etc/contrast/contrast_security.yaml`



NOTE

The `agent.service` section of the YAML configuration file only applies to earlier versions of Python (before version 5.19.0).

```
#
=====
==
# Use the properties in this YAML file to configure a Contrast agent.
# Go to https://docs.contrastsecurity.com/en/order-of-precedence.html
# to determine the order of precedence for configuration values.
#
```

```
=====
==

# Use this setting if you want to temporarily disable a Contrast agent.
# Set to `true` to enable the agent; set to `false` to disable the agent.
# enable: true

#
=====
==
# api
# Use the properties in this section to connect the agent to the Contrast
UI.
#
=====
==
api:

# ***** REQUIRED *****
# Set the URL for the Contrast UI.
url: https://app.contrastsecurity.com/Contrast

# ***** REQUIRED *****
# Set the API key needed to communicate with the Contrast UI.
api_key: NEEDS_TO_BE_SET

# ***** REQUIRED *****
# Set the service key needed to communicate with the Contrast
# UI. It is used to calculate the Authorization header.
service_key: NEEDS_TO_BE_SET

# ***** REQUIRED *****
# Set the user name used to communicate with the Contrast
# UI. It is used to calculate the Authorization header.
user_name: NEEDS_TO_BE_SET

# base64 encoded JSON object containing the `url`,
# `api_key`, `service_key`, and `user_name` config options,
# allowing them all to be set in a single variable.
# token: NEEDS_TO_BE_SET

#
=====
# api.certificate
# Use the following properties for communication
# with the Contrast UI using certificates.
#
=====
# certificate:

# If set to `false`, the agent will ignore the
# certificate configuration in this section.
# enable: true
```

```
# Set the absolute or relative path to a CA for communication
# with the Contrast UI using a self-signed certificate.
# ca_file: NEEDS_TO_BE_SET

# Set the absolute or relative path to the Certificate
# PEM file for communication with the Contrast UI.
# cert_file: NEEDS_TO_BE_SET

# Set the absolute or relative path to the Key PEM
# file for communication with the Contrast UI.
# key_file: NEEDS_TO_BE_SET

#
=====
# api.proxy
# Use the following properties for communication
# with the Contrast UI over a proxy.
#
=====
# proxy:

# Set value to `true` for the agent to communicate with
# the Contrast web interface over a proxy. Set value to
# `false` if you don't want to use the proxy. If no value is
# indicated, the presence of a valid **contrast.proxy.host**
# and **contrast.proxy.port** will enable the proxy.
# enable: NEEDS_TO_BE_SET

# Set the URL for your Proxy Server. The URL form is `scheme://
host:port`.
# url: NEEDS_TO_BE_SET

#
=====
==
# agent
# Use the properties in this section to control the way and frequency
# with which the agent communicates to logs and the Contrast UI.
#
=====
==
# agent:

#
=====
# agent.route_coverage
# Use the following properties for the route-based coverage feature.
#
=====
# route_coverage:

# Set to `false` to stop the agent from sending
# route-based coverage data to the Contrast UI when the
#
# application returns an error code indicating
```

```
# the request was not processed as expected.
#
# report_on_error: false

#
=====
# agent.logger
# Define the following properties to set logging values.
# If the following properties are not defined, the
# agent uses the logging values from the Contrast UI.
#
=====
# logger:

# Enable diagnostic logging by setting a path to a log file.
# While diagnostic logging hurts performance, it generates
# useful information for debugging Contrast. The value set here
# is the location to which the agent saves log output. If no
# log file exists at this location, the agent creates a file.
#
# Example - `/opt/Contrast/contrast.log` creates a log in the
# `/opt/Contrast` directory, and rotates it automatically as needed.
#
# path: ./contrast_agent.log

# Set the the log output level. Valid options are
# `ERROR`, `WARN`, `INFO`, `DEBUG`, and `TRACE`.
# level: INFO

# Override the name of the process the agents uses in logs.
# progname: Contrast Agent

# Set to `true` to redirect all logs to
# `stdout` instead of the file system.
# stdout: false

# Set to `true` to redirect all logs to `stderr` instead
# of the file system. Overridden by `stdout` configuration.
# stderr: false

#
=====
# agent.security_logger
# Define the following properties to set security logging
# values associated with Protect. If not defined, the agent
# uses the security logging (CEF) values from the Contrast UI.
#
=====
# security_logger:

# Set the file to which the agent logs security events.
# path: ./contrast/security.log

# Set the log level for security logging. Valid options
# are `ERROR`, `WARN`, `INFO`, `DEBUG`, and `TRACE`.
```

```
# level: ERROR

# Change the Contrast security logger from a file-sized based rolling
# scheme to a date-based rolling scheme. At midnight server time,
# the log from the previous day is renamed to *file_name.yyyy-MM-dd*.
# Note - this scheme does not have a size limit; manual log
# pruning will be required. This flag must be set to use the
# backups and size flags. Value options are `true` or `false`.
# roll_daily: NEEDS_TO_BE_SET

# Specify the file size cap (in MB) of each log file.
# roll_size: NEEDS_TO_BE_SET

# Specify the number of backup logs that the agent will create before
# Contrast cleans up the oldest file. A value of `0` means that no
backups
# are created, and the log is truncated when it reaches its size cap.
#
# Note - this property must be used with
# `agent.security_logger.roll_daily=false`; otherwise,
# Contrast continues to log daily and disregard this limit.
#
# backups: NEEDS_TO_BE_SET

#
=====
# agent.security_logger.syslog
# Define the following properties to set Syslog values. If the
properties
# are not defined, the agent uses the Syslog values from the Contrast
UI.
#
=====
# syslog:

# Set to `true` to enable Syslog logging.
# enable: NEEDS_TO_BE_SET

# Set the IP address of the Syslog server
# to which the agent should send messages.
# ip: NEEDS_TO_BE_SET

# Set the port of the Syslog server to
# which the agent should send messages.
# port: NEEDS_TO_BE_SET

# Set the facility code of the messages the agent sends to Syslog.
# facility: 19

# Set the log level of Exploited attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_exploited: ALERT

# Set the log level of Blocked attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
```

```
# severity_blocked: NOTICE

# Set the log level of Blocked At Perimeter
# attacks. Value options are `ALERT`, `CRITICAL`,
# `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_blocked_perimeter: NOTICE

# Set the log level of Probed attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_probed: WARNING

# Set the log level of Suspicious attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_suspicious: WARNING

#
=====
# agent.python
# The following properties apply to any Python agent-wide configurations.
#
=====
# python:

# Allow the agent to dump `cProfile` data to file for each request.
# enable_profiler: false

#
=====
==
# inventory
# Use the properties in this section to override the inventory features.
#
=====
==
# inventory:

# Set to `false` to disable inventory features in the agent.
# enable: true

# Set to `false` to disable library analysis.
# analyze_libraries: true

# Apply a list of labels to libraries. Labels
# must be formatted as a comma-delimited list.
# Example - `label1, label2, label3`
#
# tags: NEEDS_TO_BE_SET

#
=====
==
# assess
# Use the properties in this section to control Assess.
#
=====
```

```
==
# assess:

# Include this property to determine if the Assess
# feature should be enabled. If this property is not
# present, the decision is delegated to the Contrast UI.
# enable: false

# Apply a list of labels to vulnerabilities and preflight
# messages. Labels must be formatted as a comma-delimited list.
# Example - `label1, label2, label3`
#
# tags: NEEDS_TO_BE_SET

# Value options are `ALL`, `SOME`, or `NONE`.
# stacktraces: ALL

#
=====
# assess.sampling
# Use the following properties to control sampling in the agent.
#
=====
# sampling:

# Set to `true` to enable sampling.
# enable: false

# This property indicates the number of requests
# to analyze in each window before sampling begins.
# baseline: 5

# This property indicates that every *nth*
# request after the baseline is analyzed.
# request_frequency: 10

# This property indicates the duration for which a sample set is valid.
# window_ms: 180_000

#
=====
# assess.rules
# Use the following properties to control simple rule configurations.
#
=====
# rules:

# Define a list of Assess rules to disable in the agent. To view a
# list of rule names, in Contrast go to user menu > Policy Management >
# Assess rules. The rules must be formatted as a comma-delimited list.
#
# Example - Set `reflected-xss,sql-injection` to disable
# the reflected-xss rule and the sql-injection rule.
#
# disabled_rules: NEEDS_TO_BE_SET
```

```
#
=====
==
# protect
# Use the properties in this section to override Protect features.
#
=====
==
# protect:

# Include this property to determine if the Protect
# feature should be enabled. If this property is not
# present, the decision is delegated to the Contrast UI.
# enable: false

#
=====
# protect.rules
# Use the following properties to set simple rule configurations.
#
=====
# rules:

# Define a list of Protect rules to disable in the agent. To view a
# list of rule names, in Contrast go to user menu > Policy Management >
# Protect rules. The rules must be formatted as a comma-delimited list.
# disabled_rules: NEEDS_TO_BE_SET

#
=====
# protect.rules.bot-blocker
# Use the following selection to configure if the
# agent blocks bots. Set to `true` to enable blocking.
#
=====
# bot-blocker:

# Set to `true` for the agent to block known bots.
# enable: false

#
=====
# protect.rules.sql-injection
# Use the following settings to configure the sql-injection rule.
#
=====
# sql-injection:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or off.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
```



```
# mode: off

#
=====
# protect.rules.cmd-injection
# Use the following properties to configure
# how the command injection rule works.
#
=====
# cmd-injection:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.path-traversal
# Use the following properties to configure
# how the path traversal rule works.
#
=====
# path-traversal:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

# Detect when custom code attempts to access sensitive
# system files. The agent blocks if blocking is enabled.
# detect_custom_code_accessing_system_files: true

# Detect when users attempt to bypass filters by
# using "::$DATA" channels or null bytes in file
# names. The agent blocks if blocking is enabled.
# detect_common_file_exploits: true

#
=====
# protect.rules.method-tampering
# Use the following properties to configure
# how the method tampering rule works.
#
=====
# method-tampering:

# Set the mode of the rule. Value options are
```

```
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.reflected-xss
# Use the following properties to configure how
# the reflected cross-site scripting rule works.
#
=====
# reflected-xss:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.xxe
# Use the following properties to configure
# how the XML external entity works.
#
=====
# xxe:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
==
# application
# Use the properties in this section for
# the application(s) hosting this agent.
#
=====
==
# application:

# Override the reported application name.
#
# Note - On systems where multiple, distinct applications may be served
```

```
# by a single process, this configuration causes the agent to report
# all discovered applications as one application with the given name.
#
# Note - On Java systems where multiple, distinct applications may be
# served by a single process, this configuration causes the agent to
report
# all discovered applications as one application with the given name.
#
# name: NEEDS_TO_BE_SET

# Override the reported application path.
# path: NEEDS_TO_BE_SET

# Add the name of the application group with which this
# application should be associated in the Contrast UI.
# group: NEEDS_TO_BE_SET

# Add the application code this application should use in the Contrast UI.
# code: NEEDS_TO_BE_SET

# Override the reported application version.
# version: NEEDS_TO_BE_SET

# Apply labels to an application. Labels must
# be formatted as a comma-delimited list.
# Example - `label1,label2,label3`
#
# tags: NEEDS_TO_BE_SET

# Define a set of `key=value` pairs (which conforms to RFC 2253) for
# specifying user-defined metadata associated with the application. The
# set must be formatted as a comma-delimited list of `key=value` pairs.
# Example - `business-unit=accounting, office=Baltimore`
#
# metadata: NEEDS_TO_BE_SET

# Provide the ID of a session which already exists in the Contrast
# UI. Vulnerabilities discovered by the agent are associated with
# this session. If an invalid ID is supplied, the agent will be
# disabled. This option and `application.session_metadata` are
# mutually exclusive; if both are set, the agent will be disabled.
# session_id: NEEDS_TO_BE_SET

# Provide metadata which is used to create a new session ID in the
# Contrast UI. Vulnerabilities discovered by the agent are associated with
# this new session. This value should be formatted as `key=value` pairs
# (conforming to RFC 2253). Available key names for this configuration
# are branchName, buildNumber, commitHash, committer, gitTag, repository,
# testRun, and version. This option and `application.session_id` are
# mutually exclusive; if both are set the agent will be disabled.
# session_metadata: NEEDS_TO_BE_SET

#
=====
==
```

```
# server
# Use the settings in this section to set metadata for the server
# hosting this agent. Contrast recognizes common, supported server
# names, paths, types and environments. Doing this may require a new
# server or license, and it may affect functionality of some features.
#
=====
==
# server:

# Override the reported server name.
# name: localhost

# Override the reported server path.
# path: NEEDS_TO_BE_SET

# Override the reported server type.
# type: NEEDS_TO_BE_SET

# Set the environment directly to override the default set
# by the Contrast UI. This allows the user to configure the
# environment dynamically at startup rather than manually
# updating the Server in the Contrast UI themselves afterwards.
#
# Valid values include `QA`, `PRODUCTION` and `DEVELOPMENT`.
# For example, `PRODUCTION` registers this Server as
# running in a `PRODUCTION` environment, regardless of the
# organization's default environment in the Contrast UI.
#
# environment: NEEDS_TO_BE_SET

# Apply a list of labels to the server. Labels
# must be formatted as a comma-delimited list.
# Example - `label1,label2,label3`
#
# tags: NEEDS_TO_BE_SET

# Set to `false` to disable detection of cloud
# provider metadata such as resource identifiers.
# discover_cloud_resource: true

#
=====
==
# Use the properties in this YAML file to configure a Contrast agent.
# Go to https://docs.contrastsecurity.com/en/order-of-precedence.html
# to determine the order of precedence for configuration values.
#
=====
==

# Use this setting if you want to temporarily disable a Contrast agent.
# Set to `true` to enable the agent; set to `false` to disable the agent.
# enable: true
```

```
#
=====
==
# api
# Use the properties in this section to connect the agent to the Contrast
# UI.
#
=====
==
api:

# ***** REQUIRED *****
# Set the URL for the Contrast UI.
url: https://app.contrastsecurity.com/Contrast

# ***** REQUIRED *****
# Set the API key needed to communicate with the Contrast UI.
api_key: NEEDS_TO_BE_SET

# ***** REQUIRED *****
# Set the service key needed to communicate with the Contrast
# UI. It is used to calculate the Authorization header.
service_key: NEEDS_TO_BE_SET

# ***** REQUIRED *****
# Set the user name used to communicate with the Contrast
# UI. It is used to calculate the Authorization header.
user_name: NEEDS_TO_BE_SET

# base64 encoded JSON object containing the `url`,
# `api_key`, `service_key`, and `user_name` config options,
# allowing them all to be set in a single variable.
# token: NEEDS_TO_BE_SET

#
=====
# api.certificate
# Use the following properties for communication
# with the Contrast UI using certificates.
#
=====
# certificate:

# If set to `false`, the agent will ignore the
# certificate configuration in this section.
# enable: true

# Set the absolute or relative path to a CA for communication
# with the Contrast UI using a self-signed certificate.
# ca_file: NEEDS_TO_BE_SET

# Set the absolute or relative path to the Certificate
# PEM file for communication with the Contrast UI.
# cert_file: NEEDS_TO_BE_SET
```

```
# Set the absolute or relative path to the Key PEM
# file for communication with the Contrast UI.
# key_file: NEEDS_TO_BE_SET

#
=====
# api.proxy
# Use the following properties for communication
# with the Contrast UI over a proxy.
#
=====
# proxy:

# Set value to `true` for the agent to communicate with
# the Contrast web interface over a proxy. Set value to
# `false` if you don't want to use the proxy. If no value is
# indicated, the presence of a valid **contrast.proxy.host**
# and **contrast.proxy.port** will enable the proxy.
# enable: NEEDS_TO_BE_SET

# Set the URL for your Proxy Server. The URL form is `scheme://
host:port`.
# url: NEEDS_TO_BE_SET

#
=====
==
# agent
# Use the properties in this section to control the way and frequency
# with which the agent communicates to logs and the Contrast UI.
#
=====
==
# agent:

#
=====
# agent.route_coverage
# Use the following properties for the route-based coverage feature.
#
=====
# route_coverage:

# Set to `false` to stop the agent from sending
# route-based coverage data to the Contrast UI when the
#
# application returns an error code indicating
# the request was not processed as expected.
#
# report_on_error: false

#
=====
# agent.logger
```

```
# Define the following properties to set logging values.
# If the following properties are not defined, the
# agent uses the logging values from the Contrast UI.
#
=====
# logger:

# Enable diagnostic logging by setting a path to a log file.
# While diagnostic logging hurts performance, it generates
# useful information for debugging Contrast. The value set here
# is the location to which the agent saves log output. If no
# log file exists at this location, the agent creates a file.
#
# Example - `/opt/Contrast/contrast.log` creates a log in the
# `/opt/Contrast` directory, and rotates it automatically as needed.
#
# path: ./contrast_agent.log

# Set the the log output level. Valid options are
# `ERROR`, `WARN`, `INFO`, `DEBUG`, and `TRACE`.
# level: INFO

# Override the name of the process the agents uses in logs.
# progname: Contrast Agent

# Set to `true` to redirect all logs to
# `stdout` instead of the file system.
# stdout: false

# Set to `true` to redirect all logs to `stderr` instead
# of the file system. Overriden by `stdout` configuration.
# stderr: false

#
=====
# agent.security_logger
# Define the following properties to set security logging
# values associated with Protect. If not defined, the agent
# uses the security logging (CEF) values from the Contrast UI.
#
=====
# security_logger:

# Set the file to which the agent logs security events.
# path: ./contrast/security.log

# Set the log level for security logging. Valid options
# are `ERROR`, `WARN`, `INFO`, `DEBUG`, and `TRACE`.
# level: ERROR

# Change the Contrast security logger from a file-sized based rolling
# scheme to a date-based rolling scheme. At midnight server time,
# the log from the previous day is renamed to *file_name.yyyy-MM-dd*.
# Note - this scheme does not have a size limit; manual log
# pruning will be required. This flag must be set to use the
```

```
# backups and size flags. Value options are `true` or `false`.
# roll_daily: NEEDS_TO_BE_SET

# Specify the file size cap (in MB) of each log file.
# roll_size: NEEDS_TO_BE_SET

# Specify the number of backup logs that the agent will create before
# Contrast cleans up the oldest file. A value of `0` means that no
backups
# are created, and the log is truncated when it reaches its size cap.
#
# Note - this property must be used with
# `agent.security_logger.roll_daily=false`; otherwise,
# Contrast continues to log daily and disregard this limit.
#
# backups: NEEDS_TO_BE_SET

#
=====
# agent.security_logger.syslog
# Define the following properties to set Syslog values. If the
properties
# are not defined, the agent uses the Syslog values from the Contrast
UI.
#
=====
# syslog:

# Set to `true` to enable Syslog logging.
# enable: NEEDS_TO_BE_SET

# Set the IP address of the Syslog server
# to which the agent should send messages.
# ip: NEEDS_TO_BE_SET

# Set the port of the Syslog server to
# which the agent should send messages.
# port: NEEDS_TO_BE_SET

# Set the facility code of the messages the agent sends to Syslog.
# facility: 19

# Set the log level of Exploited attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_exploited: ALERT

# Set the log level of Blocked attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_blocked: NOTICE

# Set the log level of Blocked At Perimeter
# attacks. Value options are `ALERT`, `CRITICAL`,
# `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_blocked_perimeter: NOTICE
```



```
# Set the log level of Probed attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_probed: WARNING

# Set the log level of Suspicious attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_suspicious: WARNING

#
=====
# agent.python
# The following properties apply to any Python agent-wide configurations.
#
=====
# python:

# Allow the agent to dump `cProfile` data to file for each request.
# enable_profiler: false

#
=====
==
# inventory
# Use the properties in this section to override the inventory features.
#
=====
==
# inventory:

# Set to `false` to disable inventory features in the agent.
# enable: true

# Set to `false` to disable library analysis.
# analyze_libraries: true

# Apply a list of labels to libraries. Labels
# must be formatted as a comma-delimited list.
# Example - `label1, label2, label3`
#
# tags: NEEDS_TO_BE_SET

#
=====
==
# assess
# Use the properties in this section to control Assess.
#
=====
==
# assess:

# Include this property to determine if the Assess
# feature should be enabled. If this property is not
# present, the decision is delegated to the Contrast UI.
# enable: false
```

```
# Apply a list of labels to vulnerabilities and preflight
# messages. Labels must be formatted as a comma-delimited list.
# Example - `label1, label2, label3`
#
# tags: NEEDS_TO_BE_SET

# Value options are `ALL`, `SOME`, or `NONE`.
# stacktraces: ALL

#
=====
# assess.sampling
# Use the following properties to control sampling in the agent.
#
=====
# sampling:

# Set to `true` to enable sampling.
# enable: false

# This property indicates the number of requests
# to analyze in each window before sampling begins.
# baseline: 5

# This property indicates that every *nth*
# request after the baseline is analyzed.
# request_frequency: 10

# This property indicates the duration for which a sample set is valid.
# window_ms: 180_000

#
=====
# assess.rules
# Use the following properties to control simple rule configurations.
#
=====
# rules:

# Define a list of Assess rules to disable in the agent. To view a
# list of rule names, in Contrast go to user menu > Policy Management >
# Assess rules. The rules must be formatted as a comma-delimited list.
#
# Example - Set `reflected-xss,sql-injection` to disable
# the reflected-xss rule and the sql-injection rule.
#
# disabled_rules: NEEDS_TO_BE_SET

#
=====
==
# protect
# Use the properties in this section to override Protect features.
#
```

```
=====
==
# protect:

# Include this property to determine if the Protect
# feature should be enabled. If this property is not
# present, the decision is delegated to the Contrast UI.
# enable: false

#
=====
# protect.rules
# Use the following properties to set simple rule configurations.
#
=====
# rules:

# Define a list of Protect rules to disable in the agent. To view a
# list of rule names, in Contrast go to user menu > Policy Management >
# Protect rules. The rules must be formatted as a comma-delimited list.
# disabled_rules: NEEDS_TO_BE_SET

#
=====
# protect.rules.bot-blocker
# Use the following selection to configure if the
# agent blocks bots. Set to `true` to enable blocking.
#
=====
# bot-blocker:

# Set to `true` for the agent to block known bots.
# enable: false

#
=====
# protect.rules.sql-injection
# Use the following settings to configure the sql-injection rule.
#
=====
# sql-injection:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or off.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.cmd-injection
# Use the following properties to configure
# how the command injection rule works.
```

```
#
=====
# cmd-injection:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.path-traversal
# Use the following properties to configure
# how the path traversal rule works.
#
=====
# path-traversal:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

# Detect when custom code attempts to access sensitive
# system files. The agent blocks if blocking is enabled.
# detect_custom_code_accessing_system_files: true

# Detect when users attempt to bypass filters by
# using "::$DATA" channels or null bytes in file
# names. The agent blocks if blocking is enabled.
# detect_common_file_exploits: true

#
=====
# protect.rules.method-tampering
# Use the following properties to configure
# how the method tampering rule works.
#
=====
# method-tampering:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off
```

```
#
=====
# protect.rules.reflected-xss
# Use the following properties to configure how
# the reflected cross-site scripting rule works.
#
=====
# reflected-xss:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.xxe
# Use the following properties to configure
# how the XML external entity works.
#
=====
# xxe:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
==
# application
# Use the properties in this section for
# the application(s) hosting this agent.
#
=====
==
# application:

# Override the reported application name.
#
# Note - On systems where multiple, distinct applications may be served
# by a single process, this configuration causes the agent to report
# all discovered applications as one application with the given name.
#
# Note - On Java systems where multiple, distinct applications may be
# served by a single process, this configuration causes the agent to
report
# all discovered applications as one application with the given name.
```

```
#
# name: NEEDS_TO_BE_SET

# Override the reported application path.
# path: NEEDS_TO_BE_SET

# Add the name of the application group with which this
# application should be associated in the Contrast UI.
# group: NEEDS_TO_BE_SET

# Add the application code this application should use in the Contrast UI.
# code: NEEDS_TO_BE_SET

# Override the reported application version.
# version: NEEDS_TO_BE_SET

# Apply labels to an application. Labels must
# be formatted as a comma-delimited list.
# Example - `label1,label2,label3`
#
# tags: NEEDS_TO_BE_SET

# Define a set of `key=value` pairs (which conforms to RFC 2253) for
# specifying user-defined metadata associated with the application. The
# set must be formatted as a comma-delimited list of `key=value` pairs.
# Example - `business-unit=accounting, office=Baltimore`
#
# metadata: NEEDS_TO_BE_SET

# Provide the ID of a session which already exists in the Contrast
# UI. Vulnerabilities discovered by the agent are associated with
# this session. If an invalid ID is supplied, the agent will be
# disabled. This option and `application.session_metadata` are
# mutually exclusive; if both are set, the agent will be disabled.
# session_id: NEEDS_TO_BE_SET

# Provide metadata which is used to create a new session ID in the
# Contrast UI. Vulnerabilities discovered by the agent are associated with
# this new session. This value should be formatted as `key=value` pairs
# (conforming to RFC 2253). Available key names for this configuration
# are branchName, buildNumber, commitHash, committer, gitTag, repository,
# testRun, and version. This option and `application.session_id` are
# mutually exclusive; if both are set the agent will be disabled.
# session_metadata: NEEDS_TO_BE_SET

#
=====
==
# server
# Use the settings in this section to set metadata for the server
# hosting this agent. Contrast recognizes common, supported server
# names, paths, types and environments. Doing this may require a new
# server or license, and it may affect functionality of some features.
#
=====
```

```
==
# server:

# Override the reported server name.
# name: localhost

# Override the reported server path.
# path: NEEDS_TO_BE_SET

# Override the reported server type.
# type: NEEDS_TO_BE_SET

# Set the environment directly to override the default set
# by the Contrast UI. This allows the user to configure the
# environment dynamically at startup rather than manually
# updating the Server in the Contrast UI themselves afterwards.
#
# Valid values include `QA`, `PRODUCTION` and `DEVELOPMENT`.
# For example, `PRODUCTION` registers this Server as
# running in a `PRODUCTION` environment, regardless of the
# organization's default environment in the Contrast UI.
#
# environment: NEEDS_TO_BE_SET

# Apply a list of labels to the server. Labels
# must be formatted as a comma-delimited list.
# Example - `label1,label2,label3`
#
# tags: NEEDS_TO_BE_SET

# Set to `false` to disable detection of cloud
# provider metadata such as resource identifiers.
# discover_cloud_resource: true
```

Validate the Python agent configuration

The Python agent includes a script for validating and testing your agent configuration.

Steps to run the script

1. Install contrast-agent version 5.1.0 or later using `pip`.
2. Prepare your Contrast configuration using a [YAML file \(page 413\)](#), [environment variables \(page 110\)](#), or both.
3. Run the command `contrast-validate-config`. This script validates your current configuration and tests your ability to communicate with Contrast.

**NOTE**

A **Response: 400 Bad Request** message still indicates a successful connection to Contrast.

For example, this indicates that the application Server has not yet been created in Contrast:

```
[contrast-validate-config] Sending test request to Contrast UI
[contrast-validate-config] Response: 400 Bad Request
[contrast-validate-config] {
  "success" : false,
  "messages" : [ "Invalid agent request" ]
}
[contrast-validate-config] 400 status code indicates success
for this endpoint
[contrast-validate-config] Connection to the Contrast UI
successful
```

And this indicates that the Server has already been created in Contrast:

```
[contrast-validate-config] Sending test request to Contrast UI
[contrast-validate-config] Response: 304 Not Modified
[contrast-validate-config] 304 status code indicates success
for this endpoint
[contrast-validate-config] Connection to the Contrast UI
successful
```

See also

- [Configure the Python agent \(page 413\)](#)

Configure middleware**WARNING**

Manual middleware configuration is no longer recommended. Instead, you should run your application with the `contrast-python-run` command, which automatically detects and enables framework-specific instrumentation. See [Contrast Runner \(page 467\)](#) for more information.

For rarer cases where manual middleware configuration is still required, instructions are listed below.

Middleware is a software component that is part of a web application and is capable of reading and modifying inbound requests and outbound responses.

The Python agent is implemented as a middleware for all of the frameworks that Contrast supports. To use the Python agent, you must configure the middleware for the framework that your application uses:

AIOHTTP

AIOHTTP middleware is a class-based middleware that should be passed into the aiohttp `web.Application` constructor as an argument. The following example shows a sample AIOHTTP application that uses the Contrast middleware class:

```
from aiohttp import web
from contrast.aiohttp import ContrastMiddleware

routes = web.RouteTableDef()

@routes.get("/")
def index(request):
    raise web.HTTPFound("/hello")

middlewares = [ContrastMiddleware(app_name="app name")]
app = web.Application(middlewares=middlewares)
app.add_routes(routes)

web.run_app(app)
```

Bottle

Contrast's Bottle middleware is a WSGI middleware, which operates by wrapping the Bottle application instance.

To configure the Python agent for Bottle:

1. Find the Bottle application object in your application codebase. This will be an instance of `bottle.Bottle`.
A sample Bottle application might look like this, where `app` is your Bottle application object:

```
from bottle import Bottle, run

app = Bottle(__name__)

@app.route("/")
def index():
    return "hello world"

<InstallContrastHere>

run(app)
```

2. Wrap the Bottle application object with Contrast's middleware. In the example above, replace `<InstallContrastHere>` with:

```
from contrast.bottle import ContrastMiddleware
app = ContrastMiddleware(app)
```



IMPORTANT

In rare cases, you may be required to pass the original `bottle.Bottle` application instance directly to the `ContrastMiddleware` in addition to your WSGI application object. If the application does not start, find your original Bottle application and pass it to Contrast's middleware. For example:

```
app = ContrastMiddleware(  
    app,  
    original_bottle_app, # instance of bottle.Bottle  
)
```

Django

New configuration steps: works with Contrast Python agent versions 4.6.0 and later

Contrast's Django middleware is a WSGI middleware, not a Django-style middleware.

1. Find your WSGI application object. The `WSGI_APPLICATION` Django configuration option points to your project's WSGI app - this is often located in `wsgi.py`. You must set the `WSGI_APPLICATION` config option if it is not already set.

A sample `wsgi.py` might look like this, where `application` is your WSGI application object:

```
from django.core.wsgi import get_wsgi_application  
application = get_wsgi_application()
```

<InstallContrastHere>

2. Wrap the WSGI application object with Contrast's middleware. In the example above, replace <InstallContrastHere> with:

```
from contrast.django import ContrastMiddleware  
application = ContrastMiddleware(application)
```

Old configuration steps: will not work with Contrast Python agent versions 5.0.0 and later

Django middleware is configured in the `settings.py` file.

1. **Find the `settings.py` file.** This file isn't found in the same location for all applications, but it's generally near the top of the application source tree. Common locations include:

- `/settings.py`
- `config/settings.py`
- `app/settings.py`



NOTE

When searching the source tree to find the `settings.py`, make sure to exclude any directories that correspond to Python virtual environments.

Some applications have multiple `settings.py` files, which may correspond to different configurations of the application (for example, prod or test). In these cases, add the Contrast agent middleware to any and all of the configurations where it will be used.

2. **Add the Contrast agent module to the list.**

Include the Contrast middleware as early in the list as possible; although modifying the order may be necessary to get the application working in some circumstances.

- **Django 1.10 and later:** Look for the `MIDDLEWARE` configuration variable, which is an array. Add the Contrast agent module to the list:

```
MIDDLEWARE = [  
    'contrast.agent.middlewares.django_middleware.DjangoMiddleware',  
    # OTHER MIDDLEWARE,  
]
```

- **Django 1.6 to 1.9:** Look for the `MIDDLEWARE_CLASSES` configuration variable in *settings.py* and add the Contrast agent module to the list:

```
MIDDLEWARE_CLASSES = [  
  
    'contrast.agent.middlewares.legacy_django_middleware.DjangoMiddleware',  
    # OTHER MIDDLEWARE  
]
```

See the [Django documentation](#) for more details on middleware inclusion.

Falcon (ASGI)

Falcon (ASGI) middleware is an ASGI middleware, which operates by wrapping the Falcon(ASGI) application instance.

The following example shows a sample Falcon(ASGI) application wrapped by the Contrast middleware class.



NOTE

If your Falcon (ASGI) application uses other middleware in addition to Contrast, Contrast must be the first middleware initialized in order for certain features to work.

```
import falcon.asgi  
from contrast.falcon_asgi import ContrastMiddleware  
  
class Home(object):  
    async def on_get(self, req, resp):  
        resp.status = falcon.HTTP_200  
        resp.body = "Hello World"  
  
def create():  
    # This is where the example app is declared. Look for something similar  
    in your  
    # application since this instance needs to be wrapped by Contrast  
    middleware  
    _app = falcon.asgi.App()  
  
    # Add routes to your app  
    home = Home()  
    _app.add_route("/home", home)  
  
    # This line wraps the application instance with the Contrast middleware  
    # NOTE: Contrast should be the first middleware if others are used  
    _app = ContrastMiddleware(_app)
```

```
return _app

app = create()
```

Falcon (WSGI)

Contrast's Falcon middleware is a WSGI middleware, not a Falcon-style middleware.

1. Find your Falcon application object. This will be an instance of `falcon.API` or `falcon.App` depending on your version of Falcon.

A sample Falcon application might look like this, where `app` is your Falcon application object:

```
import falcon
from views import Home

app = falcon.API()

home = Home()
app.add_route('/home', home)

<InstallContrastHere>
```

2. Wrap the Falcon application object with Contrast's middleware. In the example above, replace `<InstallContrastHere>` with:

```
from contrast.falcon import ContrastMiddleware
app = ContrastMiddleware(app)
```



IMPORTANT

In rare cases, you may be required to pass the original `falcon.App` or `falcon.API` application instance directly to the `ContrastMiddleware` in addition to your WSGI application object. If the application does not start, find your original Falcon application and pass it to Contrast's middleware. For example:

```
app = ContrastMiddleware(
    app,
    original_falcon_app, # instance of falcon.App or
    falcon.API
)
```



IMPORTANT

You must wrap the Falcon instance *after* route registration is complete.

FastAPI

FastAPI middleware is a class-based ASGI middleware that relies on `starlette.middleware.BaseHTTPMiddleware`. Check [Python supported technologies \(page 410\)](#) for the latest version of FastAPI supported by Contrast. The following example shows a sample FastAPI application that uses the Contrast middleware class:

```
from fastapi import FastAPI
from contrast.fastapi import ContrastMiddleware

app = FastAPI()
```

```
app.add_middleware(ContrastMiddleware, original_app=app)

@app.get("/")
def read_root():
    return RedirectResponse("/home")
```

If your FastAPI application uses other middleware in addition to Contrast, Contrast must be the first middleware initialized in order for certain features to work.

```
from fastapi import FastAPI
from fastapi.middleware.httpsredirect import HTTPSRedirectMiddleware
from contrast.fastapi import ContrastMiddleware

app = FastAPI()

# ContrastMiddleware must be the first middleware
app.add_middleware(ContrastMiddleware, original_app=app)
app.add_middleware(HTTPSRedirectMiddleware)
```

Adding middleware via the `add_middleware` method is the only supported way to add middleware at this time, because certain features require the FastAPI `app` to be passed in as the `original_app` keyword argument. Initializing middlewares by passing them in directly to the FastAPI class initialization is not currently supported by Contrast.

```
# Not currently supported.
app = FastAPI(middleware=[...])
```



WARNING

Calling `add_middleware` multiple times is known to **re-initialize all previous middleware**.

Flask

Contrast's Flask middleware is a WSGI middleware, which operates by wrapping the Flask application instance.

1. Find your Flask application object. This will be an instance of `flask.Flask`.
A sample Flask application might look like this, where `app` is your Flask application object:

```
import Flask

app = Flask(__name__)

<InstallContrastHere>

@app.route('/')
def index():
    return render_template('index.html')

app.run(...)
```

2. Wrap the Flask application object with Contrast's middleware. In the example above, replace `<InstallContrastHere>` with:

```
from contrast.flask import ContrastMiddleware
app.wsgi_app = ContrastMiddleware(app)
```

**NOTE**

Contrast's Flask middleware requires an instance of `flask.Flask` as an argument, rather than `flask.Flask.wsgi_app`.

Pyramid

Old configuration steps: will not work with Contrast Python agent versions 5.0.0 and later

In Pyramid, middlewares are called "tweens".

1. Find the configurator object in your application codebase. It might look like this:

```
from pyramid.config import Configurator
config = Configurator()

<InstallContrastHere>
```

2. Add Contrast's middleware to your config. In the example above, replace `<InstallContrastHere>` with:

```
config.add_tween('contrast.agent.middlewares.pyramid_middleware.PyramidMiddleware')
```

See the [Pyramid documentation](#) for additional details on tween configuration.

New configuration steps: works with Contrast Python agent versions 4.6.0 and later

Contrast's Pyramid middleware is a WSGI middleware, not a Pyramid-style "tween".

1. Find your WSGI application object. This is often created by a call to `Configurator.make_wsgi_app()`. For example, it might look like this:

```
from pyramid.config import Configurator

config = Configurator()
app = config.make_wsgi_app()

<InstallContrastHere>
```

2. Wrap the WSGI application object with Contrast's middleware. In the example above, replace `<InstallContrastHere>` with:

```
from contrast.pyramid import ContrastMiddleware
app = ContrastMiddleware(app)
```

**IMPORTANT**

In rare cases, Contrast may require that you pass your application's Registry object to the middleware as well. The registry is commonly available as an attribute of both the Configurator instance and the Pyramid application object.

If the application does not start, find your application Registry and pass it to Contrast's middleware, for example:

```
app = ContrastMiddleware(app, config.registry)
```

Quart

The Quart middleware should wrap the Quart application object and be assigned to the `asgi_app` attribute. The following example shows a sample quart application that uses the Contrast middleware class:

```
from quart import Quart, redirect
from contrast.quart import ContrastMiddleware

app = Quart(__name__)
app.asgi_app = ContrastMiddleware(app)

@app.route("/")
async def index():
    return redirect("/home")
```

WSGI

Contrast provides a generic WSGI middleware that includes all core agent functionality. Framework-specific features such as route discovery are not implemented in the generic WSGI middleware.

To instrument any WSGI-compliant application with Contrast:

1. Find the WSGI application object in your application codebase. For example, a WSGI app was created as follows:

```
from example.module import make_app
wsgi_app = make_app()

<InstallContrastHere>
```

2. Wrap the WSGI application object with Contrast's middleware. In the example above, replace `<InstallContrastHere>` with:

```
from contrast.wsgi import ContrastMiddleware
wsgi_app = ContrastMiddleware(wsgi_app)
```

See the [WSGI specification](#) for additional details on WSGI middleware.

ASGI

Contrast also provides a generic ASGI middleware. Like the generic WSGI middleware, the ASGI middleware includes core agent functionality but no framework-specific features.

To instrument an ASGI-compliant HTTP/HTTPS application:

1. Find the ASGI application object in your codebase. For example, take the following ASGI app:

```
from example.module import make_app
asgi_app = make_app()

<InstallContrastHere>
```

2. Wrap the ASGI application object with Contrast's middleware. In the example above, replace `<InstallContrastHere>` with:

```
from contrast.asgi import ContrastMiddleware
asgi_app = ContrastMiddleware(asgi_app)
```

Python web servers

The Python agent is tested against several common production web servers. Some servers require certain configuration options in order to work properly with Contrast.

Gunicorn

You can integrate Gunicorn with popular frameworks like Django, Pyramid, Tornado, or others.

Use the following command to start Gunicorn: `gunicorn app.wsgi_app:application` (where `app` is the folder of your application).

The following configurations are available within your Gunicorn server:

- **Bind:**

```
-b ADDRESS1
gunicorn -b 127.0.0.1:8000
```

- **Timeout:**

```
-t INT or --timeout INT
```

Workers silent for more than this many seconds are killed and restarted.

Value is a positive number or 0. Setting it to 0 has the effect of infinite timeouts by disabling timeouts for all workers entirely.

The default is 30 seconds. Only set this noticeably higher if you're sure of the repercussions for sync workers. For the non sync workers, it means that the worker process is still communicating and is not tied to the length of time required to handle a single request.

Uvicorn

Uvicorn is an asynchronous web server that can be installed with: `pip install uvicorn[standard]`.

The following are some suggested configurations:

- `--loop=uvloop`

- `--http=httptools`

Both `uvloop` and `httptools` are written in Cython, and offer greater performance but are not compatible with Windows or PyPy.

- `--interface`

You may set this to some of the ASGI protocols. If you are using WSGI, and using `ws=websockets`, `websockets` will not be used and defaults to `auto`.

- `--ws`

If you have set `--ws-max-size`, `--ws-ping-interval`, `--ws-ping-timeout`, and `--ws` is not set with `websockets`, everything will be ignored.

- To use gunicorn as a manager to uvicorn you can do that by the following:

```
gunicorn -k uvicorn.workers.UvicornWorker
```

- The [UvicornWorker implementation](#) uses the `uvloop` and `httptools` implementations. To run under PyPy, you will need to use pure-python implementation instead. You can do this by using the `UvicornH11Worker` class. `gunicorn -k uvicorn.workers.UvicornH11Worker`.

uWSGI configuration

Contrast requires the following configuration options when running on uWSGI. You can set these on the command line or in your uWSGI configuration (.ini) file:

- `--enable-threads`: Contrast's machinery utilizes threads. This option must be enabled so the agent can start background threads.
- `--single-interpreter`: The Python agent applies its instrumentation to the Python process in which it is initialized. This option ensures that Contrast is initialized in the same process that handles requests for your application.
- If using `--master` you must also use `--lazy-apps`: When running in master mode, uWSGI initializes the application in a master process but forks this process into workers which handle requests. To operate correctly, Contrast must be initialized independently in each worker process; `--lazy-apps` achieves this.

Python YAML template

Use this template to configure the Python agent using a YAML configuration file. (Learn more about [YAML configuration \(page 107\)](#).)

Place your YAML file in the default location: `/etc/contrast/contrast_security.yaml`



NOTE

The `agent.service` section of the YAML configuration file only applies to earlier versions of Python (before version 5.19.0).

```
#
=====
==
# Use the properties in this YAML file to configure a Contrast agent.
# Go to https://docs.contrastsecurity.com/en/order-of-precedence.html
# to determine the order of precedence for configuration values.
#
=====
==

# Use this setting if you want to temporarily disable a Contrast agent.
# Set to `true` to enable the agent; set to `false` to disable the agent.
# enable: true

#
=====
==
# api
# Use the properties in this section to connect the agent to the Contrast
UI.
#
=====
==
api:

# ***** REQUIRED *****
```

```
# Set the URL for the Contrast UI.
url: https://app.contrastsecurity.com/Contrast

# ***** REQUIRED *****
# Set the API key needed to communicate with the Contrast UI.
api_key: NEEDS_TO_BE_SET

# ***** REQUIRED *****
# Set the service key needed to communicate with the Contrast
# UI. It is used to calculate the Authorization header.
service_key: NEEDS_TO_BE_SET

# ***** REQUIRED *****
# Set the user name used to communicate with the Contrast
# UI. It is used to calculate the Authorization header.
user_name: NEEDS_TO_BE_SET

# base64 encoded JSON object containing the `url`,
# `api_key`, `service_key`, and `user_name` config options,
# allowing them all to be set in a single variable.
# token: NEEDS_TO_BE_SET
```

```
#
```

```
=====
# api.certificate
# Use the following properties for communication
# with the Contrast UI using certificates.
#
```

```
=====
# certificate:
```

```
    # If set to `false`, the agent will ignore the
    # certificate configuration in this section.
    # enable: true
```

```
    # Set the absolute or relative path to a CA for communication
    # with the Contrast UI using a self-signed certificate.
    # ca_file: NEEDS_TO_BE_SET
```

```
    # Set the absolute or relative path to the Certificate
    # PEM file for communication with the Contrast UI.
    # cert_file: NEEDS_TO_BE_SET
```

```
    # Set the absolute or relative path to the Key PEM
    # file for communication with the Contrast UI.
    # key_file: NEEDS_TO_BE_SET
```

```
#
```

```
=====
# api.proxy
# Use the following properties for communication
# with the Contrast UI over a proxy.
#
```

```
=====
# proxy:
```

```
# Set value to `true` for the agent to communicate with
# the Contrast web interface over a proxy. Set value to
# `false` if you don't want to use the proxy. If no value is
# indicated, the presence of a valid **contrast.proxy.host**
# and **contrast.proxy.port** will enable the proxy.
# enable: NEEDS_TO_BE_SET

# Set the URL for your Proxy Server. The URL form is `scheme://
host:port`.
# url: NEEDS_TO_BE_SET

#
=====
==
# agent
# Use the properties in this section to control the way and frequency
# with which the agent communicates to logs and the Contrast UI.
#
=====
==
# agent:

#
=====
# agent.route_coverage
# Use the following properties for the route-based coverage feature.
#
=====
# route_coverage:

# Set to `false` to stop the agent from sending
# route-based coverage data to the Contrast UI when the
#
# application returns an error code indicating
# the request was not processed as expected.
#
# report_on_error: false

#
=====
# agent.logger
# Define the following properties to set logging values.
# If the following properties are not defined, the
# agent uses the logging values from the Contrast UI.
#
=====
# logger:

# Enable diagnostic logging by setting a path to a log file.
# While diagnostic logging hurts performance, it generates
# useful information for debugging Contrast. The value set here
# is the location to which the agent saves log output. If no
# log file exists at this location, the agent creates a file.
#
```

```
# Example - `/opt/Contrast/contrast.log` creates a log in the
# `/opt/Contrast` directory, and rotates it automatically as needed.
#
# path: ./contrast_agent.log

# Set the the log output level. Valid options are
# `ERROR`, `WARN`, `INFO`, `DEBUG`, and `TRACE`.
# level: INFO

# Override the name of the process the agents uses in logs.
# progname: Contrast Agent

# Set to `true` to redirect all logs to
# `stdout` instead of the file system.
# stdout: false

# Set to `true` to redirect all logs to `stderr` instead
# of the file system. Overridden by `stdout` configuration.
# stderr: false

#
=====
# agent.security_logger
# Define the following properties to set security logging
# values associated with Protect. If not defined, the agent
# uses the security logging (CEF) values from the Contrast UI.
#
=====
# security_logger:

# Set the file to which the agent logs security events.
# path: ./contrast/security.log

# Set the log level for security logging. Valid options
# are `ERROR`, `WARN`, `INFO`, `DEBUG`, and `TRACE`.
# level: ERROR

# Change the Contrast security logger from a file-sized based rolling
# scheme to a date-based rolling scheme. At midnight server time,
# the log from the previous day is renamed to *file_name.yyyy-MM-dd*.
# Note - this scheme does not have a size limit; manual log
# pruning will be required. This flag must be set to use the
# backups and size flags. Value options are `true` or `false`.
# roll_daily: NEEDS_TO_BE_SET

# Specify the file size cap (in MB) of each log file.
# roll_size: NEEDS_TO_BE_SET

# Specify the number of backup logs that the agent will create before
# Contrast cleans up the oldest file. A value of `0` means that no
backups
# are created, and the log is truncated when it reaches its size cap.
#
# Note - this property must be used with
# `agent.security_logger.roll_daily=false`; otherwise,
```

```
# Contrast continues to log daily and disregard this limit.
#
# backups: NEEDS_TO_BE_SET

#
=====
# agent.security_logger.syslog
# Define the following properties to set Syslog values. If the
properties
# are not defined, the agent uses the Syslog values from the Contrast
UI.
#
=====
# syslog:

# Set to `true` to enable Syslog logging.
# enable: NEEDS_TO_BE_SET

# Set the IP address of the Syslog server
# to which the agent should send messages.
# ip: NEEDS_TO_BE_SET

# Set the port of the Syslog server to
# which the agent should send messages.
# port: NEEDS_TO_BE_SET

# Set the facility code of the messages the agent sends to Syslog.
# facility: 19

# Set the log level of Exploited attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_exploited: ALERT

# Set the log level of Blocked attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_blocked: NOTICE

# Set the log level of Blocked At Perimeter
# attacks. Value options are `ALERT`, `CRITICAL`,
# `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_blocked_perimeter: NOTICE

# Set the log level of Probed attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_probed: WARNING

# Set the log level of Suspicious attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_suspicious: WARNING

#
=====
# agent.python
# The following properties apply to any Python agent-wide configurations.
#
```

```
=====
# python:

# Allow the agent to dump `cProfile` data to file for each request.
# enable_profiler: false

#
=====
==
# inventory
# Use the properties in this section to override the inventory features.
#
=====
==
# inventory:

# Set to `false` to disable inventory features in the agent.
# enable: true

# Set to `false` to disable library analysis.
# analyze_libraries: true

# Apply a list of labels to libraries. Labels
# must be formatted as a comma-delimited list.
# Example - `label1, label2, label3`
#
# tags: NEEDS_TO_BE_SET

#
=====
==
# assess
# Use the properties in this section to control Assess.
#
=====
==
# assess:

# Include this property to determine if the Assess
# feature should be enabled. If this property is not
# present, the decision is delegated to the Contrast UI.
# enable: false

# Apply a list of labels to vulnerabilities and preflight
# messages. Labels must be formatted as a comma-delimited list.
# Example - `label1, label2, label3`
#
# tags: NEEDS_TO_BE_SET

# Value options are `ALL`, `SOME`, or `NONE`.
# stacktraces: ALL

#
=====
# assess.sampling
```

```
# Use the following properties to control sampling in the agent.
#
=====
# sampling:

# Set to `true` to enable sampling.
# enable: false

# This property indicates the number of requests
# to analyze in each window before sampling begins.
# baseline: 5

# This property indicates that every *nth*
# request after the baseline is analyzed.
# request_frequency: 10

# This property indicates the duration for which a sample set is valid.
# window_ms: 180_000

#
=====
# assess.rules
# Use the following properties to control simple rule configurations.
#
=====
# rules:

# Define a list of Assess rules to disable in the agent. To view a
# list of rule names, in Contrast go to user menu > Policy Management >
# Assess rules. The rules must be formatted as a comma-delimited list.
#
# Example - Set `reflected-xss,sql-injection` to disable
# the reflected-xss rule and the sql-injection rule.
#
# disabled_rules: NEEDS_TO_BE_SET

#
=====
==
# protect
# Use the properties in this section to override Protect features.
#
=====
==
# protect:

# Include this property to determine if the Protect
# feature should be enabled. If this property is not
# present, the decision is delegated to the Contrast UI.
# enable: false

#
=====
# protect.rules
# Use the following properties to set simple rule configurations.
```

```
#
=====
# rules:

# Define a list of Protect rules to disable in the agent. To view a
# list of rule names, in Contrast go to user menu > Policy Management >
# Protect rules. The rules must be formatted as a comma-delimited list.
# disabled_rules: NEEDS_TO_BE_SET

#
=====
# protect.rules.bot-blocker
# Use the following selection to configure if the
# agent blocks bots. Set to `true` to enable blocking.
#
=====
# bot-blocker:

# Set to `true` for the agent to block known bots.
# enable: false

#
=====
# protect.rules.sql-injection
# Use the following settings to configure the sql-injection rule.
#
=====
# sql-injection:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or off.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.cmd-injection
# Use the following properties to configure
# how the command injection rule works.
#
=====
# cmd-injection:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
```



```
=====
# protect.rules.path-traversal
# Use the following properties to configure
# how the path traversal rule works.
#
=====
# path-traversal:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

# Detect when custom code attempts to access sensitive
# system files. The agent blocks if blocking is enabled.
# detect_custom_code_accessing_system_files: true

# Detect when users attempt to bypass filters by
# using "::$DATA" channels or null bytes in file
# names. The agent blocks if blocking is enabled.
# detect_common_file_exploits: true

#
=====
# protect.rules.method-tampering
# Use the following properties to configure
# how the method tampering rule works.
#
=====
# method-tampering:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.reflected-xss
# Use the following properties to configure how
# the reflected cross-site scripting rule works.
#
=====
# reflected-xss:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
```

```
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.xxe
# Use the following properties to configure
# how the XML external entity works.
#
=====
# xxe:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
==
# application
# Use the properties in this section for
# the application(s) hosting this agent.
#
=====
==
# application:

# Override the reported application name.
#
# Note - On systems where multiple, distinct applications may be served
# by a single process, this configuration causes the agent to report
# all discovered applications as one application with the given name.
#
# Note - On Java systems where multiple, distinct applications may be
# served by a single process, this configuration causes the agent to
report
# all discovered applications as one application with the given name.
#
# name: NEEDS_TO_BE_SET

# Override the reported application path.
# path: NEEDS_TO_BE_SET

# Add the name of the application group with which this
# application should be associated in the Contrast UI.
# group: NEEDS_TO_BE_SET

# Add the application code this application should use in the Contrast UI.
# code: NEEDS_TO_BE_SET
```

```
# Override the reported application version.
# version: NEEDS_TO_BE_SET

# Apply labels to an application. Labels must
# be formatted as a comma-delimited list.
# Example - `label1,label2,label3`
#
# tags: NEEDS_TO_BE_SET

# Define a set of `key=value` pairs (which conforms to RFC 2253) for
# specifying user-defined metadata associated with the application. The
# set must be formatted as a comma-delimited list of `key=value` pairs.
# Example - `business-unit=accounting, office=Baltimore`
#
# metadata: NEEDS_TO_BE_SET

# Provide the ID of a session which already exists in the Contrast
# UI. Vulnerabilities discovered by the agent are associated with
# this session. If an invalid ID is supplied, the agent will be
# disabled. This option and `application.session_metadata` are
# mutually exclusive; if both are set, the agent will be disabled.
# session_id: NEEDS_TO_BE_SET

# Provide metadata which is used to create a new session ID in the
# Contrast UI. Vulnerabilities discovered by the agent are associated with
# this new session. This value should be formatted as `key=value` pairs
# (conforming to RFC 2253). Available key names for this configuration
# are branchName, buildNumber, commitHash, committer, gitTag, repository,
# testRun, and version. This option and `application.session_id` are
# mutually exclusive; if both are set the agent will be disabled.
# session_metadata: NEEDS_TO_BE_SET

#
=====
==
# server
# Use the settings in this section to set metadata for the server
# hosting this agent. Contrast recognizes common, supported server
# names, paths, types and environments. Doing this may require a new
# server or license, and it may affect functionality of some features.
#
=====
==
# server:

# Override the reported server name.
# name: localhost

# Override the reported server path.
# path: NEEDS_TO_BE_SET

# Override the reported server type.
# type: NEEDS_TO_BE_SET

# Set the environment directly to override the default set
```

```
# by the Contrast UI. This allows the user to configure the
# environment dynamically at startup rather than manually
# updating the Server in the Contrast UI themselves afterwards.
#
# Valid values include `QA`, `PRODUCTION` and `DEVELOPMENT`.
# For example, `PRODUCTION` registers this Server as
# running in a `PRODUCTION` environment, regardless of the
# organization's default environment in the Contrast UI.
#
# environment: NEEDS_TO_BE_SET

# Apply a list of labels to the server. Labels
# must be formatted as a comma-delimited list.
# Example - `label1,label2,label3`
#
# tags: NEEDS_TO_BE_SET

# Set to `false` to disable detection of cloud
# provider metadata such as resource identifiers.
# discover_cloud_resource: true
```

```
#
=====
==
# Use the properties in this YAML file to configure a Contrast agent.
# Go to https://docs.contrastsecurity.com/en/order-of-precedence.html
# to determine the order of precedence for configuration values.
#
=====
==

# Use this setting if you want to temporarily disable a Contrast agent.
# Set to `true` to enable the agent; set to `false` to disable the agent.
# enable: true

#
=====
==
# api
# Use the properties in this section to connect the agent to the Contrast
# UI.
#
=====
==
api:

# ***** REQUIRED *****
# Set the URL for the Contrast UI.
url: https://app.contrastsecurity.com/Contrast

# ***** REQUIRED *****
# Set the API key needed to communicate with the Contrast UI.
api_key: NEEDS_TO_BE_SET
```

```
# ***** REQUIRED *****
# Set the service key needed to communicate with the Contrast
# UI. It is used to calculate the Authorization header.
service_key: NEEDS_TO_BE_SET

# ***** REQUIRED *****
# Set the user name used to communicate with the Contrast
# UI. It is used to calculate the Authorization header.
user_name: NEEDS_TO_BE_SET

# base64 encoded JSON object containing the `url`,
# `api_key`, `service_key`, and `user_name` config options,
# allowing them all to be set in a single variable.
# token: NEEDS_TO_BE_SET

#
=====
# api.certificate
# Use the following properties for communication
# with the Contrast UI using certificates.
#
=====
# certificate:

# If set to `false`, the agent will ignore the
# certificate configuration in this section.
# enable: true

# Set the absolute or relative path to a CA for communication
# with the Contrast UI using a self-signed certificate.
# ca_file: NEEDS_TO_BE_SET

# Set the absolute or relative path to the Certificate
# PEM file for communication with the Contrast UI.
# cert_file: NEEDS_TO_BE_SET

# Set the absolute or relative path to the Key PEM
# file for communication with the Contrast UI.
# key_file: NEEDS_TO_BE_SET

#
=====
# api.proxy
# Use the following properties for communication
# with the Contrast UI over a proxy.
#
=====
# proxy:

# Set value to `true` for the agent to communicate with
# the Contrast web interface over a proxy. Set value to
# `false` if you don't want to use the proxy. If no value is
# indicated, the presence of a valid contrast.proxy.host
# and contrast.proxy.port will enable the proxy.
# enable: NEEDS_TO_BE_SET
```

```
# Set the URL for your Proxy Server. The URL form is `scheme://
host:port`.
# url: NEEDS_TO_BE_SET

#
=====
==
# agent
# Use the properties in this section to control the way and frequency
# with which the agent communicates to logs and the Contrast UI.
#
=====
==
# agent:

#
=====
# agent.route_coverage
# Use the following properties for the route-based coverage feature.
#
=====
# route_coverage:

# Set to `false` to stop the agent from sending
# route-based coverage data to the Contrast UI when the
#
# application returns an error code indicating
# the request was not processed as expected.
#
# report_on_error: false

#
=====
# agent.logger
# Define the following properties to set logging values.
# If the following properties are not defined, the
# agent uses the logging values from the Contrast UI.
#
=====
# logger:

# Enable diagnostic logging by setting a path to a log file.
# While diagnostic logging hurts performance, it generates
# useful information for debugging Contrast. The value set here
# is the location to which the agent saves log output. If no
# log file exists at this location, the agent creates a file.
#
# Example - `/opt/Contrast/contrast.log` creates a log in the
# `/opt/Contrast` directory, and rotates it automatically as needed.
#
# path: ./contrast_agent.log

# Set the the log output level. Valid options are
# `ERROR`, `WARN`, `INFO`, `DEBUG`, and `TRACE`.
```

```
# level: INFO

# Override the name of the process the agents uses in logs.
# progname: Contrast Agent

# Set to `true` to redirect all logs to
# `stdout` instead of the file system.
# stdout: false

# Set to `true` to redirect all logs to `stderr` instead
# of the file system. Overridden by `stdout` configuration.
# stderr: false

#
=====
# agent.security_logger
# Define the following properties to set security logging
# values associated with Protect. If not defined, the agent
# uses the security logging (CEF) values from the Contrast UI.
#
=====
# security_logger:

# Set the file to which the agent logs security events.
# path: ./contrast/security.log

# Set the log level for security logging. Valid options
# are `ERROR`, `WARN`, `INFO`, `DEBUG`, and `TRACE`.
# level: ERROR

# Change the Contrast security logger from a file-sized based rolling
# scheme to a date-based rolling scheme. At midnight server time,
# the log from the previous day is renamed to *file_name.yyyy-MM-dd*.
# Note - this scheme does not have a size limit; manual log
# pruning will be required. This flag must be set to use the
# backups and size flags. Value options are `true` or `false`.
# roll_daily: NEEDS_TO_BE_SET

# Specify the file size cap (in MB) of each log file.
# roll_size: NEEDS_TO_BE_SET

# Specify the number of backup logs that the agent will create before
# Contrast cleans up the oldest file. A value of `0` means that no
backups
# are created, and the log is truncated when it reaches its size cap.
#
# Note - this property must be used with
# `agent.security_logger.roll_daily=false`; otherwise,
# Contrast continues to log daily and disregard this limit.
#
# backups: NEEDS_TO_BE_SET

#
=====
# agent.security_logger.syslog
```

```
# Define the following properties to set Syslog values. If the
properties
# are not defined, the agent uses the Syslog values from the Contrast
UI.
#
=====
# syslog:

# Set to `true` to enable Syslog logging.
# enable: NEEDS_TO_BE_SET

# Set the IP address of the Syslog server
# to which the agent should send messages.
# ip: NEEDS_TO_BE_SET

# Set the port of the Syslog server to
# which the agent should send messages.
# port: NEEDS_TO_BE_SET

# Set the facility code of the messages the agent sends to Syslog.
# facility: 19

# Set the log level of Exploited attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_exploited: ALERT

# Set the log level of Blocked attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_blocked: NOTICE

# Set the log level of Blocked At Perimeter
# attacks. Value options are `ALERT`, `CRITICAL`,
# `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_blocked_perimeter: NOTICE

# Set the log level of Probed attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_probed: WARNING

# Set the log level of Suspicious attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_suspicious: WARNING

#
=====
# agent.python
# The following properties apply to any Python agent-wide configurations.
#
=====
# python:

# Allow the agent to dump `cProfile` data to file for each request.
# enable_profiler: false

#
```



```
=====
==
# inventory
# Use the properties in this section to override the inventory features.
#
=====
==
# inventory:

# Set to `false` to disable inventory features in the agent.
# enable: true

# Set to `false` to disable library analysis.
# analyze_libraries: true

# Apply a list of labels to libraries. Labels
# must be formatted as a comma-delimited list.
# Example - `label1, label2, label3`
#
# tags: NEEDS_TO_BE_SET

#
=====
==
# assess
# Use the properties in this section to control Assess.
#
=====
==
# assess:

# Include this property to determine if the Assess
# feature should be enabled. If this property is not
# present, the decision is delegated to the Contrast UI.
# enable: false

# Apply a list of labels to vulnerabilities and preflight
# messages. Labels must be formatted as a comma-delimited list.
# Example - `label1, label2, label3`
#
# tags: NEEDS_TO_BE_SET

# Value options are `ALL`, `SOME`, or `NONE`.
# stacktraces: ALL

#
=====
# assess.sampling
# Use the following properties to control sampling in the agent.
#
=====
# sampling:

# Set to `true` to enable sampling.
# enable: false
```

```
# This property indicates the number of requests
# to analyze in each window before sampling begins.
# baseline: 5

# This property indicates that every *nth*
# request after the baseline is analyzed.
# request_frequency: 10

# This property indicates the duration for which a sample set is valid.
# window_ms: 180_000

#
=====
# assess.rules
# Use the following properties to control simple rule configurations.
#
=====
# rules:

# Define a list of Assess rules to disable in the agent. To view a
# list of rule names, in Contrast go to user menu > Policy Management >
# Assess rules. The rules must be formatted as a comma-delimited list.
#
# Example - Set `reflected-xss,sql-injection` to disable
# the reflected-xss rule and the sql-injection rule.
#
# disabled_rules: NEEDS_TO_BE_SET

#
=====
==
# protect
# Use the properties in this section to override Protect features.
#
=====
==
# protect:

# Include this property to determine if the Protect
# feature should be enabled. If this property is not
# present, the decision is delegated to the Contrast UI.
# enable: false

#
=====
# protect.rules
# Use the following properties to set simple rule configurations.
#
=====
# rules:

# Define a list of Protect rules to disable in the agent. To view a
# list of rule names, in Contrast go to user menu > Policy Management >
# Protect rules. The rules must be formatted as a comma-delimited list.
```

```
# disabled_rules: NEEDS_TO_BE_SET

#
=====
# protect.rules.bot-blocker
# Use the following selection to configure if the
# agent blocks bots. Set to `true` to enable blocking.
#
=====
# bot-blocker:

# Set to `true` for the agent to block known bots.
# enable: false

#
=====
# protect.rules.sql-injection
# Use the following settings to configure the sql-injection rule.
#
=====
# sql-injection:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or off.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.cmd-injection
# Use the following properties to configure
# how the command injection rule works.
#
=====
# cmd-injection:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.path-traversal
# Use the following properties to configure
# how the path traversal rule works.
#
=====
# path-traversal:
```

```
# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

# Detect when custom code attempts to access sensitive
# system files. The agent blocks if blocking is enabled.
# detect_custom_code_accessing_system_files: true

# Detect when users attempt to bypass filters by
# using "::$DATA" channels or null bytes in file
# names. The agent blocks if blocking is enabled.
# detect_common_file_exploits: true

#
=====
# protect.rules.method-tampering
# Use the following properties to configure
# how the method tampering rule works.
#
=====
# method-tampering:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.reflected-xss
# Use the following properties to configure how
# the reflected cross-site scripting rule works.
#
=====
# reflected-xss:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.xxe
```

```
# Use the following properties to configure
# how the XML external entity works.
#
=====
# xxe:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
==
# application
# Use the properties in this section for
# the application(s) hosting this agent.
#
=====
==
# application:

# Override the reported application name.
#
# Note - On systems where multiple, distinct applications may be served
# by a single process, this configuration causes the agent to report
# all discovered applications as one application with the given name.
#
# Note - On Java systems where multiple, distinct applications may be
# served by a single process, this configuration causes the agent to
report
# all discovered applications as one application with the given name.
#
# name: NEEDS_TO_BE_SET

# Override the reported application path.
# path: NEEDS_TO_BE_SET

# Add the name of the application group with which this
# application should be associated in the Contrast UI.
# group: NEEDS_TO_BE_SET

# Add the application code this application should use in the Contrast UI.
# code: NEEDS_TO_BE_SET

# Override the reported application version.
# version: NEEDS_TO_BE_SET

# Apply labels to an application. Labels must
# be formatted as a comma-delimited list.
# Example - `label1,label2,label3`
#
```

```
# tags: NEEDS_TO_BE_SET

# Define a set of `key=value` pairs (which conforms to RFC 2253) for
# specifying user-defined metadata associated with the application. The
# set must be formatted as a comma-delimited list of `key=value` pairs.
# Example - `business-unit=accounting, office=Baltimore`
#
# metadata: NEEDS_TO_BE_SET

# Provide the ID of a session which already exists in the Contrast
# UI. Vulnerabilities discovered by the agent are associated with
# this session. If an invalid ID is supplied, the agent will be
# disabled. This option and `application.session_metadata` are
# mutually exclusive; if both are set, the agent will be disabled.
# session_id: NEEDS_TO_BE_SET

# Provide metadata which is used to create a new session ID in the
# Contrast UI. Vulnerabilities discovered by the agent are associated with
# this new session. This value should be formatted as `key=value` pairs
# (conforming to RFC 2253). Available key names for this configuration
# are branchName, buildNumber, commitHash, committer, gitTag, repository,
# testRun, and version. This option and `application.session_id` are
# mutually exclusive; if both are set the agent will be disabled.
# session_metadata: NEEDS_TO_BE_SET

#
=====
==
# server
# Use the settings in this section to set metadata for the server
# hosting this agent. Contrast recognizes common, supported server
# names, paths, types and environments. Doing this may require a new
# server or license, and it may affect functionality of some features.
#
=====
==
# server:

# Override the reported server name.
# name: localhost

# Override the reported server path.
# path: NEEDS_TO_BE_SET

# Override the reported server type.
# type: NEEDS_TO_BE_SET

# Set the environment directly to override the default set
# by the Contrast UI. This allows the user to configure the
# environment dynamically at startup rather than manually
# updating the Server in the Contrast UI themselves afterwards.
#
# Valid values include `QA`, `PRODUCTION` and `DEVELOPMENT`.
# For example, `PRODUCTION` registers this Server as
# running in a `PRODUCTION` environment, regardless of the
```

```
# organization's default environment in the Contrast UI.
#
# environment: NEEDS_TO_BE_SET

# Apply a list of labels to the server. Labels
# must be formatted as a comma-delimited list.
# Example - `label1,label2,label3`
#
# tags: NEEDS_TO_BE_SET

# Set to `false` to disable detection of cloud
# provider metadata such as resource identifiers.
# discover_cloud_resource: true
```

Python telemetry

The Python agent use telemetry to collect usage data. Telemetry is collected when an instrumented application first loads the agent's sensors and then periodically (every few hours) afterwards.

[Your privacy is important to us \(page 1275\)](#). The telemetry feature does not collect application data. The data is anonymized before being sent securely to Contrast. Then the aggregated data is stored encrypted and under restricted access control. Any collected data will be deleted after one year.

The telemetry feature collects the following data:

Agent versions	Data
Python 4.14.0	Agent version
	Operating system and version
	Python version
	Application framework and version
	Web server and version
	Hosted or on-premises Contrast instance

To opt-out of the telemetry feature, set the `CONTRAST_AGENT_TELEMETRY_OPTOUT` environment variable to `1` or `true`.

Telemetry data is securely sent to `telemetry.python.contrastsecurity.com`. You can also opt out of telemetry by blocking communication at the network level.

Contrast Runner

As of version 5.24.0, the Python agent provides a new command-line interface (also called a runner) for instrumenting Python applications. In general, when using the runner, it is no longer required to [manually configure middleware \(page 436\)](#). Instead, the Contrast runner automatically applies framework-specific instrumentation to your Python application.



NOTE

Instrumentation tools that use a runner or wrapper script may conflict with the Contrast Runner. If your application has issues starting when you use the Contrast Runner with other instrumentation tools, [configure your application with Contrast's middleware \(page 436\)](#).

The runner command is called `contrast-python-run` and is provided as part of the `contrast-agent` package. In most Python environments it will be available on the command line without further changes when the `contrast-agent` package is [installed \(page 411\)](#).

Using the runner

The runner is used by adding the `contrast-python-run` command to the beginning of the original command that starts your application.

- For example, with a **Django** application that is started with the following command:

```
python manage.py runserver
```

- You would run the following command to add Contrast to this application:

```
contrast-python-run -- python manage.py runserver
```



NOTE

The double dash separator “--” is used to separate the runner command's arguments from those that belong to the original command.

- In another example, with a **Flask** application that is started with the following command:

```
FLASK_APP=apps/app.py flask run --host=localhost --port=8080
```

- Using the Contrast runner would look like this (note that setting the environment variable still happens before the runner command):

```
FLASK_APP=apps/app.py contrast-python-run -- flask run --host=localhost --port=8080
```

- The `contrast-python-run` command also works when deploying with web servers such as **uwsgi** and **gunicorn**. For example:

```
contrast-python-run -- gunicorn apps/app:app --preload -b localhost:8080
```

- It also works when deploying to Apache using **mod_wsgi-express**. For example:

```
contrast-python-run -- mod_wsgi-express start-server app/wsgi.py --  
user=www-data --group www-data
```

The runner follows the normal [order of precedence \(page 106\)](#) for Contrast configuration. It is also possible to use environment variables with the runner directly on the command line:

```
CONTRAST__AGENT__LOGGER__LEVEL=DEBUG contrast-python-run -- python  
manage.py runserver
```

Ruby agent



IMPORTANT

The Ruby Agent is not currently being sold to new customers. Please contact our Sales team if you have any questions about our offerings and support for the Ruby language.

The Ruby agent is a Rack middleware that's compatible with the most popular web application frameworks. The agent's goal is to be fully Rack compatible and to provide applications built on

Rack with interactive application security testing (IAST) and runtime application self-protection (RASP) capabilities.

In Assess, the agent identifies vulnerable dataflow paths and other issues during the normal execution of your application. It reports these findings to your organization in Contrast; you can then remediate the vulnerabilities before deploying the application in a live environment.

In Protect, the Ruby agent inspects HTTP requests to identify potentially harmful input vectors. During the request, the agent inspects database queries, file writes and other potentially damaging actions resulting from the request. At the end of the request, the agent inspects the rendered output for successful attacks and can block a successful attack from being forwarded to the application user. It then sends the details of the attack to the Contrast application, which then sends you an alert and displays attack details in the interface.

**NOTE**

The Ruby agent supports Assess, Protect, and SCA.

As a next step, you can:

- [Install the Ruby agent \(page 471\)](#)
- [View Ruby agent supported technologies \(page 469\)](#)

Supported technologies for the Ruby agent

**IMPORTANT**

The Ruby Agent is not currently being sold to new customers. Please contact our Sales team if you have any questions about our offerings and support for the Ruby language.

We support the following technologies for this agent.

Technology	Supported versions	Notes
Language version	<ul style="list-style-type: none">• 3.2.X: First supported agent was 6.13.0• 3.1.X: First supported agent was 6.0.0	<p>Contrast supports Ruby Long-Term Support (LTS) versions in normal maintenance and security maintenance status. Contrast shifts its support for Ruby versions as the working group shifts its LTS windows.</p> <p>See the Ruby Maintenance Branches schedule for specific release dates.</p> <p>See Ruby end of life for dates.</p> <p>Not supported:</p> <ul style="list-style-type: none">• 2.5.X: Last supported agent was 4.14.1• 2.4.X: Last supported agent was 3.9.1• 2.6.X: Last supported agent was 5.3.0• 2.7.X: Last supported agent was 6.15.3• 3.0.X: First supported agent was 4.6.0
Application frameworks	<ul style="list-style-type: none">• Rails 7.X• Sinatra 3.X	<p>While the agent can still run on Rack-based web frameworks that aren't officially supported, Contrast may produce less-specific findings than it does for supported frameworks. Instead of reporting that a vulnerability occurs in your application code, Contrast may report it within the framework code where it interfaces directly with the Rack methods.</p>
Databases	<ul style="list-style-type: none">• MongoDB• MySQL2• PG• SQLite3	
Web servers	<ul style="list-style-type: none">• Passenger 5.37 and 6.X• Puma 3.7 - 5.X• Thin 1.7.2 - 1.8.X• Unicorn 5.0.X - 6.X	

System requirements for the Ruby agent



IMPORTANT

The Ruby Agent is not currently being sold to new customers. Please contact our Sales team if you have any questions about our offerings and support for the Ruby language.

Before installing the Ruby agent, your system must meet the following requirements:

- There is a deployed application to be analyzed, and the web application technology is supported by Contrast.
- The application can be restarted.
- The web server has network connectivity with Contrast.
- The web server has network connectivity with RubyGems or the agent manually installed.
- The server meets the minimum requirements shown in this table.

Requirement	Versions	Notes
Operating system	<ul style="list-style-type: none">64-bit Linux (recommended)64-bit Alpine	<p>Starting with agent version 3.0.0, the gem installation step requires the compilation of C extensions. This process is automatic, but you may need to install the following in the target environment:</p> <ul style="list-style-type: none"><code>gcc</code>, <code>make</code>, <code>automake</code> and <code>autoconf</code> (Package names may vary. You may install your platform's version of <code>build-essential</code>.)system headers <p>The agent runs in the Ruby application layer with some C dependencies so it may need to be installed with the <code>--platform ruby</code> flag to allow for compilation in either <i>glibc</i> or <i>musl</i> based systems.</p>
Ruby gems	<ul style="list-style-type: none">ougai: <code>>1.8</code> (1.8 and later, earlier than 2)rack: <code>>=2.0</code> (2.0 and later)	

Install the Ruby agent



IMPORTANT

The Ruby Agent is not currently being sold to new customers. Please contact our Sales team if you have any questions about our offerings and support for the Ruby language.

The `contrast-agent.gem` is a standard Ruby library that you can add to the application Gemfile.

Install the Ruby agent [using RubyGems as a gem source \(page 471\)](#) or [update the Ruby agent \(page 472\)](#).

Install the Ruby agent with RubyGems as a gem source



IMPORTANT

The Ruby Agent is not currently being sold to new customers. Please contact our Sales team if you have any questions about our offerings and support for the Ruby language.

To download the Ruby agent from RubyGems:

1. Add this to your application's gemfile:

```
gem 'contrast-agent'
```

2. Run an install:

```
bundle install
```

or an update:

```
bundle update contrast-agent
```

3. [Configure middleware \(page 473\)](#) (Rails or Sinatra).
4. [Configure the agent \(page 473\)](#).

5. Verify that `autoconf` is installed on the system where you will run the agent.
6. Verify that you can see your application in Contrast.

**NOTE**

If installing in Alpine, you may need to run the command `bundle config force_ruby_platform true` prior to installation.

**NOTE**

If you only want to run with Contrast in certain environments, you can do so using the [Bundler's Group function](#).

Ruby update agent

**IMPORTANT**

The Ruby Agent is not currently being sold to new customers. Please contact our Sales team if you have any questions about our offerings and support for the Ruby language.

The most reliable and effective way to automatically update the Contrast Ruby agent is to use the Ruby `Bundler package manager` to install and download the latest version available. Because Bundler typically manages all dependencies for your Ruby application, it should already be available and part of your build environment. How frequently you update the Contrast Ruby agent and where you get updates depends on your organization's preferences and your Contrast implementation: Hosted or on-premises.

The main steps are:

1. Choose a source for the Contrast Ruby agent.
2. Install the agent,
3. Use scripts for automatic updates.

Before you begin

- Some familiarity with Ruby's Bundler package manager.
- Access to the RubyGems repository for the Contrast Ruby agent.
- Confirmed that your Ruby application runs properly without the Contrast Ruby agent.
- Previously successfully installed the Contrast Ruby agent.
- Defined a policy for how and when to update the agent, based on your change management policy and the environment where you deploy agents.

Install and use scripts automatic updates

1. Choose a source for the Contrast ruby agent:
 - RubyGems public (or private) repository
2. Include the Contrast Ruby agent in the Gemfile to easily keep every new build of your application aligned with the latest version of the agent. Do not specify a version for contrast-agent, and it will retrieve the latest version.
The Gemfile is where you specify which dependencies you want to automatically resolve every time your Ruby application builds with artifacts from RubyGems (public or private).
3. After you update the Gemfile, use the following command when you build your application. This will automatically download and add the Contrast Ruby agent from RubyGems to the Ruby application.

```
$ bundle install
```

4. After you add contrast-agent to your Gemfile, you can use Bundler to update your agent, like this:

```
bundle update contrast-agent
```

Install the gem manually

You can manually update agents as part of a Ruby build process in two ways. Choose the one that works best for your organization and workflow:

- **Rubygems:** Use the following command to retrieve and install the Contrast Ruby agent from RubyGems (public or private) to the application:

```
$ gem install contrast-agent
```

Add the following line to your Gemfile to manage updates with Bundler, because the previous command only installs the agent locally:

```
gem "contrast-agent"
```

Then to either install or update the agent using bundler run the following:

```
$ bundle install
```

- Add the following line to your Gemfile to manage the updates with Bundler, because the previous command only installs the agent locally:

```
gem "contrast-agent"
```

See also

- [Ruby supported technologies \(page 469\)](#)
- [Install Ruby \(page 471\)](#)

Configure the Ruby agent

The [configuration \(page 102\)](#) for all agents uses this [order of precedence \(page 106\)](#). Configure the agent using a YAML configuration file.



TIP

Use the [Contrast agent configuration editor \(page 109\)](#) to create or upload a YAML configuration file, validate YAML, and get setting recommendations.

Follow these guidelines to configure your:

- [Frameworks \(page 495\)](#)
 - [Rails \(page 495\)](#)
 - [Sinatra \(page 495\)](#)
 - [Grape \(page 496\)](#)
- [Web servers \(page 498\)](#)
 - [Passenger \(page 498\)](#)
 - [Puma \(page 502\)](#)
 - [Thin \(page 504\)](#)
 - [Unicorn \(page 505\)](#)

See also

- [Ruby supported technologies \(page 469\)](#)

Ruby YAML template

Use this template to configure the Ruby agent using a YAML configuration file. (Learn more about [YAML configuration \(page 107\)](#).)

Place your YAML file in the default location: `/etc/contrast/contrast_security.yaml`

```
#
=====
==
# Use the properties in this YAML file to configure a Contrast agent.
# Go to https://docs.contrastsecurity.com/en/order-of-precedence.html
# to determine the order of precedence for configuration values.
#
=====
==

# Use this setting if you want to temporarily disable a Contrast agent.
# Set to `true` to enable the agent; set to `false` to disable the agent.
# enable: true

#
=====
==
# api
# Use the properties in this section to connect the agent to the Contrast
# UI.
#
=====
==
api:

  # ***** REQUIRED *****
  # Set the URL for the Contrast UI.
  url: https://app.contrastsecurity.com/Contrast

  # ***** REQUIRED *****
  # Set the API key needed to communicate with the Contrast UI.
  api_key: NEEDS_TO_BE_SET

  # ***** REQUIRED *****
```

```
# Set the service key needed to communicate with the Contrast
# UI. It is used to calculate the Authorization header.
service_key: NEEDS_TO_BE_SET

# ***** REQUIRED *****
# Set the user name used to communicate with the Contrast
# UI. It is used to calculate the Authorization header.
user_name: NEEDS_TO_BE_SET

#
=====
# api.certificate
# Use the following properties for communication
# with the Contrast UI using certificates.
#
=====
# certificate:

# If set to `false`, the agent will ignore the
# certificate configuration in this section.
# enable: true

# Set the absolute or relative path to a CA for communication
# with the Contrast UI using a self-signed certificate.
# ca_file: NEEDS_TO_BE_SET

# Set the absolute or relative path to the Certificate
# PEM file for communication with the Contrast UI.
# cert_file: NEEDS_TO_BE_SET

# Set the absolute or relative path to the Key PEM
# file for communication with the Contrast UI.
# key_file: NEEDS_TO_BE_SET

#
=====
# api.proxy
# Use the following properties for communication
# with the Contrast UI over a proxy.
#
=====
# proxy:

# Set value to `true` for the agent to communicate with
# the Contrast web interface over a proxy. Set value to
# `false` if you don't want to use the proxy. If no value is
# indicated, the presence of a valid **contrast.proxy.host**
# and **contrast.proxy.port** will enable the proxy.
# enable: NEEDS_TO_BE_SET

# Set the URL for your Proxy Server. The URL form is `scheme://
host:port`.
# url: NEEDS_TO_BE_SET

#
```

```
=====
==
# agent
# Use the properties in this section to control the way and frequency
# with which the agent communicates to logs and the Contrast UI.
#
=====
==
# agent:

#
=====

# agent.logger
# Define the following properties to set logging values.
# If the following properties are not defined, the
# agent uses the logging values from the Contrast UI.
#
=====
# logger:

# Enable diagnostic logging by setting a path to a log file.
# While diagnostic logging hurts performance, it generates
# useful information for debugging Contrast. The value set here
# is the location to which the agent saves log output. If no
# log file exists at this location, the agent creates a file.
#
# Example - `/opt/Contrast/contrast.log` creates a log in the
# `/opt/Contrast` directory, and rotates it automatically as needed.
#
# path: ./contrast_agent.log

# Set the the log output level. Valid options are
# `ERROR`, `WARN`, `INFO`, `DEBUG`, and `TRACE`.
# level: INFO

# Override the name of the process the agents uses in logs.
# progname: Contrast Agent

# Set to `true` for the agent to tag
# logs with `!AM!` for the metrics tool.
# metrics: true

#
=====
# agent.security_logger
# Define the following properties to set security logging
# values associated with Protect. If not defined, the agent
# uses the security logging (CEF) values from the Contrast UI.
#
=====
# security_logger:

# Set the file to which the agent logs security events.
# path: ./contrast/security.log
```



```
# Set the log level for security logging. Valid options
# are `ERROR`, `WARN`, `INFO`, `DEBUG`, and `TRACE`.
# level: ERROR

#
=====
# agent.security_logger.syslog
# Define the following properties to set Syslog values. If the
properties
# are not defined, the agent uses the Syslog values from the Contrast
UI.
#
=====
# syslog:

# Set to `true` to enable Syslog logging.
# enable: NEEDS_TO_BE_SET

# Set the IP address of the Syslog server
# to which the agent should send messages.
# ip: NEEDS_TO_BE_SET

# Set the port of the Syslog server to
# which the agent should send messages.
# port: NEEDS_TO_BE_SET

# Set the facility code of the messages the agent sends to Syslog.
# facility: 19

# Set the log level of Exploited attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_exploited: ALERT

# Set the log level of Blocked attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_blocked: NOTICE

# Set the log level of Blocked At Perimeter
# attacks. Value options are `ALERT`, `CRITICAL`,
# `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_blocked_perimeter: NOTICE

# Set the log level of Probed attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_probed: WARNING

# Set the log level of Suspicious attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_suspicious: WARNING

#
=====
# agent.heap_dump
# The following properties are used to trigger heap dumps from within
# the agent to snapshot the behavior of instrumented applications.
```

```
#
=====
# heap_dump:

# Set to `true` for the agent to automatically
# take heap dumps of the instrumented application.
# enable: false

# Set the location to which to save the heap dump files. If relative,
# the path is determined based on the process' working directory.
# path: contrast_heap_dumps

# Set the amount of time to wait, in milliseconds,
# after agent startup to begin taking heap dumps.
# delay_ms: 10_000

# Set the amount of time to wait, in milliseconds, between each heap
dump.
# window_ms: 10_000

# Set the number of heap dumps to take before disabling this feature.
# count: 5

# Set to `true` for the agent to trigger garbage collection before
# taking a heap dump to remove temporary objects from the dump.
# clean: false

#
=====
# agent.ruby
# The following properties apply to any Ruby agent-wide configurations.
#
=====
# ruby:

# Allow the agent to track frozen Objects returned by
# source methods. This configuration is on by default.
# track_frozen_sources: NEEDS_TO_BE_SET

# Allow the agent to track propagation through interpolated
# Strings. This configuration is on by default.
# interpolate: NEEDS_TO_BE_SET

# Set a comma-separated string of rake tasks
# in which to disable agent operation.
# disabled_agent_rake_tasks:
about,assets:clean,assets:clobber,assets:environment,assets:precompile,asset
s:precompile:all,db:create,db:drop,db:migrate:status,db:rollback,db:schema:c
ache:clear,db:schema:cache:dump,db:schema:dump,db:schema:load,db:seed,db:set
up,db:structure:dump,db:version,doc:app,log:clear,middleware,notes,notes:cus
tom,rails:template,rails:update,routes,secret,spec,spec:features,spec:reques
ts,spec:controllers,spec:helpers,spec:models,spec:views,spec:routing,spec:rc
ov,stats,test,test:all,test:all:db,test:recent,test:single,test:uncommitted,
time:zones:all,tmp:clear,tmp:create,webpacker:compile
```

```
#
=====
==
# inventory
# Use the properties in this section to override the inventory features.
#
=====
==
# inventory:

# Set to `false` to disable inventory features in the agent.
# enable: true

# Apply a list of labels to libraries. Labels
# must be formatted as a comma-delimited list.
# Example - `label1, label2, label3`
#
# tags: NEEDS_TO_BE_SET

#
=====
==
# assess
# Use the properties in this section to control Assess.
#
=====
==
# assess:

# Include this property to determine if the Assess
# feature should be enabled. If this property is not
# present, the decision is delegated to the Contrast UI.
# enable: false

# Apply a list of labels to vulnerabilities and preflight
# messages. Labels must be formatted as a comma-delimited list.
# Example - `label1, label2, label3`
#
# tags: NEEDS_TO_BE_SET

# Value options are `ALL`, `SOME`, or `NONE`.
# stacktraces: ALL

#
=====
# assess.sampling
# Use the following properties to control sampling in the agent.
#
=====
# sampling:

# Set to `true` to enable sampling.
# enable: false

# This property indicates the number of requests
```

```
# to analyze in each window before sampling begins.
# baseline: 5

# This property indicates that every *nth*
# request after the baseline is analyzed.
# request_frequency: 10

# This property indicates the duration for which a sample set is valid.
# window_ms: 180_000

#
=====
# assess.rules
# Use the following properties to control simple rule configurations.
#
=====
# rules:

# Define a list of Assess rules to disable in the agent. To view a
# list of rule names, in Contrast go to user menu > Policy Management >
# Assess rules. The rules must be formatted as a comma-delimited list.
#
# Example - Set `reflected-xss,sql-injection` to disable
# the reflected-xss rule and the sql-injection rule.
#
# disabled_rules: NEEDS_TO_BE_SET

#
=====
==
# protect
# Use the properties in this section to override Protect features.
#
=====
==
# protect:

# Include this property to determine if the Protect
# feature should be enabled. If this property is not
# present, the decision is delegated to the Contrast UI.
# enable: false

#
=====
# protect.rules
# Use the following properties to set simple rule configurations.
#
=====
# rules:

# Define a list of Protect rules to disable in the agent. To view a
# list of rule names, in Contrast go to user menu > Policy Management >
# Protect rules. The rules must be formatted as a comma-delimited list.
# disabled_rules: NEEDS_TO_BE_SET
```

```
#
=====
# protect.rules.bot-blocker
# Use the following selection to configure if the
# agent blocks bots. Set to `true` to enable blocking.
#
=====
# bot-blocker:

# Set to `true` for the agent to block known bots.
# enable: false

#
=====
# protect.rules.sql-injection
# Use the following settings to configure the sql-injection rule.
#
=====
# sql-injection:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or off.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.cmd-injection
# Use the following properties to configure
# how the command injection rule works.
#
=====
# cmd-injection:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.path-traversal
# Use the following properties to configure
# how the path traversal rule works.
#
=====
# path-traversal:

# Set the mode of the rule. Value options are
```

```
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.method-tampering
# Use the following properties to configure
# how the method tampering rule works.
#
=====
# method-tampering:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.reflected-xss
# Use the following properties to configure how
# the reflected cross-site scripting rule works.
#
=====
# reflected-xss:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.xxe
# Use the following properties to configure
# how the XML external entity works.
#
=====
# xxe:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
```

```
#
# mode: off

#
=====
==
# application
# Use the properties in this section for
# the application(s) hosting this agent.
#
=====
==
# application:

# Override the reported application name.
#
# Note - On systems where multiple, distinct applications may be served
# by a single process, this configuration causes the agent to report
# all discovered applications as one application with the given name.
#
# Note - On Java systems where multiple, distinct applications may be
# served by a single process, this configuration causes the agent to
report
# all discovered applications as one application with the given name.
#
# name: NEEDS_TO_BE_SET

# Override the reported application path.
# path: NEEDS_TO_BE_SET

# Add the name of the application group with which this
# application should be associated in the Contrast UI.
# group: NEEDS_TO_BE_SET

# Add the application code this application should use in the Contrast UI.
# code: NEEDS_TO_BE_SET

# Override the reported application version.
# version: NEEDS_TO_BE_SET

# Apply labels to an application. Labels must
# be formatted as a comma-delimited list.
# Example - `label1,label2,label3`
#
# tags: NEEDS_TO_BE_SET

# Define a set of `key=value` pairs (which conforms to RFC 2253) for
# specifying user-defined metadata associated with the application. The
# set must be formatted as a comma-delimited list of `key=value` pairs.
# Example - `business-unit=accounting, office=Baltimore`
#
# metadata: NEEDS_TO_BE_SET

# Provide the ID of a session which already exists in the Contrast
# UI. Vulnerabilities discovered by the agent are associated with
```

```
# this session. If an invalid ID is supplied, the agent will be
# disabled. This option and `application.session_metadata` are
# mutually exclusive; if both are set, the agent will be disabled.
# session_id: NEEDS_TO_BE_SET

# Provide metadata which is used to create a new session ID in the
# Contrast UI. Vulnerabilities discovered by the agent are associated with
# this new session. This value should be formatted as `key=value` pairs
# (conforming to RFC 2253). Available key names for this configuration
# are branchName, buildNumber, commitHash, committer, gitTag, repository,
# testRun, and version. This option and `application.session_id` are
# mutually exclusive; if both are set the agent will be disabled.
# session_metadata: NEEDS_TO_BE_SET

#
=====
==
# server
# Use the settings in this section to set metadata for the server
# hosting this agent. Contrast recognizes common, supported server
# names, paths, types and environments. Doing this may require a new
# server or license, and it may affect functionality of some features.
#
=====
==
# server:

# Override the reported server name.
# name: localhost

# Override the reported server path.
# path: NEEDS_TO_BE_SET

# Override the reported server type.
# type: NEEDS_TO_BE_SET

# Set the environment directly to override the default set
# by the Contrast UI. This allows the user to configure the
# environment dynamically at startup rather than manually
# updating the Server in the Contrast UI themselves afterwards.
#
# Valid values include `QA`, `PRODUCTION` and `DEVELOPMENT`.
# For example, `PRODUCTION` registers this Server as
# running in a `PRODUCTION` environment, regardless of the
# organization's default environment in the Contrast UI.
#
# environment: NEEDS_TO_BE_SET

# Apply a list of labels to the server. Labels
# must be formatted as a comma-delimited list.
# Example - `label1,label2,label3`
#
# tags: NEEDS_TO_BE_SET
```



```
#
=====
==
# Use the properties in this YAML file to configure a Contrast agent.
# Go to https://docs.contrastsecurity.com/en/order-of-precedence.html
# to determine the order of precedence for configuration values.
#
=====
==

# Use this setting if you want to temporarily disable a Contrast agent.
# Set to `true` to enable the agent; set to `false` to disable the agent.
# enable: true

#
=====
==
# api
# Use the properties in this section to connect the agent to the Contrast
# UI.
#
=====
==
api:

  # ***** REQUIRED *****
  # Set the URL for the Contrast UI.
  url: https://app.contrastsecurity.com/Contrast

  # ***** REQUIRED *****
  # Set the API key needed to communicate with the Contrast UI.
  api_key: NEEDS_TO_BE_SET

  # ***** REQUIRED *****
  # Set the service key needed to communicate with the Contrast
  # UI. It is used to calculate the Authorization header.
  service_key: NEEDS_TO_BE_SET

  # ***** REQUIRED *****
  # Set the user name used to communicate with the Contrast
  # UI. It is used to calculate the Authorization header.
  user_name: NEEDS_TO_BE_SET

#
=====
# api.certificate
# Use the following properties for communication
# with the Contrast UI using certificates.
#
=====
# certificate:

  # If set to `false`, the agent will ignore the
  # certificate configuration in this section.
```

```
# enable: true

# Set the absolute or relative path to a CA for communication
# with the Contrast UI using a self-signed certificate.
# ca_file: NEEDS_TO_BE_SET

# Set the absolute or relative path to the Certificate
# PEM file for communication with the Contrast UI.
# cert_file: NEEDS_TO_BE_SET

# Set the absolute or relative path to the Key PEM
# file for communication with the Contrast UI.
# key_file: NEEDS_TO_BE_SET

#
=====
# api.proxy
# Use the following properties for communication
# with the Contrast UI over a proxy.
#
=====
# proxy:

# Set value to `true` for the agent to communicate with
# the Contrast web interface over a proxy. Set value to
# `false` if you don't want to use the proxy. If no value is
# indicated, the presence of a valid **contrast.proxy.host**
# and **contrast.proxy.port** will enable the proxy.
# enable: NEEDS_TO_BE_SET

# Set the URL for your Proxy Server. The URL form is `scheme://
host:port`.
# url: NEEDS_TO_BE_SET

#
=====
==
# agent
# Use the properties in this section to control the way and frequency
# with which the agent communicates to logs and the Contrast UI.
#
=====
==
# agent:

#
=====
# agent.logger
# Define the following properties to set logging values.
# If the following properties are not defined, the
# agent uses the logging values from the Contrast UI.
#
=====
# logger:
```

```
# Enable diagnostic logging by setting a path to a log file.
# While diagnostic logging hurts performance, it generates
# useful information for debugging Contrast. The value set here
# is the location to which the agent saves log output. If no
# log file exists at this location, the agent creates a file.
#
# Example - `/opt/Contrast/contrast.log` creates a log in the
# `/opt/Contrast` directory, and rotates it automatically as needed.
#
# path: ./contrast_agent.log

# Set the the log output level. Valid options are
# `ERROR`, `WARN`, `INFO`, `DEBUG`, and `TRACE`.
# level: INFO

# Override the name of the process the agents uses in logs.
# progname: Contrast Agent

# Set to `true` for the agent to tag
# logs with `!AM!` for the metrics tool.
# metrics: true

#
=====
# agent.security_logger
# Define the following properties to set security logging
# values associated with Protect. If not defined, the agent
# uses the security logging (CEF) values from the Contrast UI.
#
=====
# security_logger:

# Set the file to which the agent logs security events.
# path: ./contrast/security.log

# Set the log level for security logging. Valid options
# are `ERROR`, `WARN`, `INFO`, `DEBUG`, and `TRACE`.
# level: ERROR

#
=====
# agent.security_logger.syslog
# Define the following properties to set Syslog values. If the
properties
# are not defined, the agent uses the Syslog values from the Contrast
UI.
#
=====
# syslog:

# Set to `true` to enable Syslog logging.
# enable: NEEDS_TO_BE_SET

# Set the IP address of the Syslog server
# to which the agent should send messages.
```

```
# ip: NEEDS_TO_BE_SET

# Set the port of the Syslog server to
# which the agent should send messages.
# port: NEEDS_TO_BE_SET

# Set the facility code of the messages the agent sends to Syslog.
# facility: 19

# Set the log level of Exploited attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_exploited: ALERT

# Set the log level of Blocked attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_blocked: NOTICE

# Set the log level of Blocked At Perimeter
# attacks. Value options are `ALERT`, `CRITICAL`,
# `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_blocked_perimeter: NOTICE

# Set the log level of Probed attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_probed: WARNING

# Set the log level of Suspicious attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_suspicious: WARNING

#
=====
# agent.heap_dump
# The following properties are used to trigger heap dumps from within
# the agent to snapshot the behavior of instrumented applications.
#
=====
# heap_dump:

# Set to `true` for the agent to automatically
# take heap dumps of the instrumented application.
# enable: false

# Set the location to which to save the heap dump files. If relative,
# the path is determined based on the process' working directory.
# path: contrast_heap_dumps

# Set the amount of time to wait, in milliseconds,
# after agent startup to begin taking heap dumps.
# delay_ms: 10_000

# Set the amount of time to wait, in milliseconds, between each heap
dump.
# window_ms: 10_000
```

```

# Set the number of heap dumps to take before disabling this feature.
# count: 5

# Set to `true` for the agent to trigger garbage collection before
# taking a heap dump to remove temporary objects from the dump.
# clean: false

#
=====
# agent.ruby
# The following properties apply to any Ruby agent-wide configurations.
#
=====
# ruby:

# Allow the agent to track frozen Objects returned by
# source methods. This configuration is on by default.
# track_frozen_sources: NEEDS_TO_BE_SET

# Allow the agent to track propagation through interpolated
# Strings. This configuration is on by default.
# interpolate: NEEDS_TO_BE_SET

# Set a comma-separated string of rake tasks
# in which to disable agent operation.
# disabled_agent_rake_tasks:
about,assets:clean,assets:clobber,assets:environment,assets:precompile,asset
s:precompile:all,db:create,db:drop,db:migrate:status,db:rollback,db:schema:c
ache:clear,db:schema:cache:dump,db:schema:dump,db:schema:load,db:seed,db:set
up,db:structure:dump,db:version,doc:app,log:clear,middleware,notes,notes:cus
tom,rails:template,rails:update,routes,secret,spec,spec:features,spec:reques
ts,spec:controllers,spec:helpers,spec:models,spec:views,spec:routing,spec:rc
ov,stats,test,test:all,test:all:db,test:recent,test:single,test:uncommitted,
time:zones:all,tmp:clear,tmp:create,webpacker:compile

#
=====
==
# inventory
# Use the properties in this section to override the inventory features.
#
=====
==
# inventory:

# Set to `false` to disable inventory features in the agent.
# enable: true

# Apply a list of labels to libraries. Labels
# must be formatted as a comma-delimited list.
# Example - `label1, label2, label3`
#
# tags: NEEDS_TO_BE_SET

#

```

```
=====
==
# assess
# Use the properties in this section to control Assess.
#
=====
==
# assess:

# Include this property to determine if the Assess
# feature should be enabled. If this property is not
# present, the decision is delegated to the Contrast UI.
# enable: false

# Apply a list of labels to vulnerabilities and preflight
# messages. Labels must be formatted as a comma-delimited list.
# Example - `label1, label2, label3`
#
# tags: NEEDS_TO_BE_SET

# Value options are `ALL`, `SOME`, or `NONE`.
# stacktraces: ALL

#
=====
# assess.sampling
# Use the following properties to control sampling in the agent.
#
=====
# sampling:

# Set to `true` to enable sampling.
# enable: false

# This property indicates the number of requests
# to analyze in each window before sampling begins.
# baseline: 5

# This property indicates that every *nth*
# request after the baseline is analyzed.
# request_frequency: 10

# This property indicates the duration for which a sample set is valid.
# window_ms: 180_000

#
=====
# assess.rules
# Use the following properties to control simple rule configurations.
#
=====
# rules:

# Define a list of Assess rules to disable in the agent. To view a
# list of rule names, in Contrast go to user menu > Policy Management >
```

```
# Assess rules. The rules must be formatted as a comma-delimited list.
#
# Example - Set `reflected-xss,sql-injection` to disable
# the reflected-xss rule and the sql-injection rule.
#
# disabled_rules: NEEDS_TO_BE_SET

#
=====
==
# protect
# Use the properties in this section to override Protect features.
#
=====
==
# protect:

# Include this property to determine if the Protect
# feature should be enabled. If this property is not
# present, the decision is delegated to the Contrast UI.
# enable: false

#
=====

# protect.rules
# Use the following properties to set simple rule configurations.
#
=====

# rules:

# Define a list of Protect rules to disable in the agent. To view a
# list of rule names, in Contrast go to user menu > Policy Management >
# Protect rules. The rules must be formatted as a comma-delimited list.
# disabled_rules: NEEDS_TO_BE_SET

#
=====

# protect.rules.bot-blocker
# Use the following selection to configure if the
# agent blocks bots. Set to `true` to enable blocking.
#
=====

# bot-blocker:

# Set to `true` for the agent to block known bots.
# enable: false

#
=====

# protect.rules.sql-injection
# Use the following settings to configure the sql-injection rule.
#
=====

# sql-injection:
```

```
# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or off.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.cmd-injection
# Use the following properties to configure
# how the command injection rule works.
#
=====
# cmd-injection:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.path-traversal
# Use the following properties to configure
# how the path traversal rule works.
#
=====
# path-traversal:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.method-tampering
# Use the following properties to configure
# how the method tampering rule works.
#
=====
# method-tampering:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
```



```
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.reflected-xss
# Use the following properties to configure how
# the reflected cross-site scripting rule works.
#
=====
# reflected-xss:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.xxe
# Use the following properties to configure
# how the XML external entity works.
#
=====
# xxe:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
==
# application
# Use the properties in this section for
# the application(s) hosting this agent.
#
=====
==
# application:

# Override the reported application name.
#
# Note - On systems where multiple, distinct applications may be served
# by a single process, this configuration causes the agent to report
# all discovered applications as one application with the given name.
#
```

```
# Note - On Java systems where multiple, distinct applications may be
# served by a single process, this configuration causes the agent to
report
# all discovered applications as one application with the given name.
#
# name: NEEDS_TO_BE_SET

# Override the reported application path.
# path: NEEDS_TO_BE_SET

# Add the name of the application group with which this
# application should be associated in the Contrast UI.
# group: NEEDS_TO_BE_SET

# Add the application code this application should use in the Contrast UI.
# code: NEEDS_TO_BE_SET

# Override the reported application version.
# version: NEEDS_TO_BE_SET

# Apply labels to an application. Labels must
# be formatted as a comma-delimited list.
# Example - `label1,label2,label3`
#
# tags: NEEDS_TO_BE_SET

# Define a set of `key=value` pairs (which conforms to RFC 2253) for
# specifying user-defined metadata associated with the application. The
# set must be formatted as a comma-delimited list of `key=value` pairs.
# Example - `business-unit=accounting, office=Baltimore`
#
# metadata: NEEDS_TO_BE_SET

# Provide the ID of a session which already exists in the Contrast
# UI. Vulnerabilities discovered by the agent are associated with
# this session. If an invalid ID is supplied, the agent will be
# disabled. This option and `application.session_metadata` are
# mutually exclusive; if both are set, the agent will be disabled.
# session_id: NEEDS_TO_BE_SET

# Provide metadata which is used to create a new session ID in the
# Contrast UI. Vulnerabilities discovered by the agent are associated with
# this new session. This value should be formatted as `key=value` pairs
# (conforming to RFC 2253). Available key names for this configuration
# are branchName, buildNumber, commitHash, committer, gitTag, repository,
# testRun, and version. This option and `application.session_id` are
# mutually exclusive; if both are set the agent will be disabled.
# session_metadata: NEEDS_TO_BE_SET

#
=====
==
# server
# Use the settings in this section to set metadata for the server
# hosting this agent. Contrast recognizes common, supported server
```

```
# names, paths, types and environments. Doing this may require a new
# server or license, and it may affect functionality of some features.
#
=====
==
# server:

# Override the reported server name.
# name: localhost

# Override the reported server path.
# path: NEEDS_TO_BE_SET

# Override the reported server type.
# type: NEEDS_TO_BE_SET

# Set the environment directly to override the default set
# by the Contrast UI. This allows the user to configure the
# environment dynamically at startup rather than manually
# updating the Server in the Contrast UI themselves afterwards.
#
# Valid values include `QA`, `PRODUCTION` and `DEVELOPMENT`.
# For example, `PRODUCTION` registers this Server as
# running in a `PRODUCTION` environment, regardless of the
# organization's default environment in the Contrast UI.
#
# environment: NEEDS_TO_BE_SET

# Apply a list of labels to the server. Labels
# must be formatted as a comma-delimited list.
# Example - `label1,label2,label3`
#
# tags: NEEDS_TO_BE_SET
```

Ruby frameworks

Application frameworks are software libraries that provide a fundamental structure to support the development of applications for a specific environment.

Follow these guidelines based on your framework:

Configure with Rails

If you are using Rails, the Ruby agent functions as a [Railtie](#) so no additional configuration is required.

Configure with Sinatra

If you are using the Sinatra framework, you must configure your application to use the Ruby agent. A simple application configured to work with the Ruby agent looks like the following example:

```
require 'sinatra'
require 'contrast-agent'

class App < Sinatra::Base
  use Contrast::Agent::Middleware, true
end
```

Configure Sinatra with config.ru

If there is no class extending, `Sinatra::Base` or `config.ru` is the default way of racking up your application with the following configuration:

`config.ru`

```
# frozen_string_literal: true

# Require Sinatra early for Framework support to detect it.
require 'sinatra'

# example app.rb, could be any file implementing Sinatra
# endpoints and logic.
#
# For instance:
#
#   # frozen_string_literal: true
#   require 'sinatra'
#
#   get '/frank-says' do
#     'Put this in your pipe & smoke it!'
#   end
#
require './app.rb'

# Contrast Agent needs to be required after sinatra.
require 'contrast-agent'

# Example for requiring gems:
require 'bundler/setup'
Bundler.require(:default)

# Add Contrast Agent middleware to the rack stack:
use Contrast::Agent::Middleware, true

# Run Sinatra application:
run Sinatra::Application
```

Start the application with:

```
bundle exec rackup
```

OR

```
bundle exec rackup config.ru
```

Configure with Grape

If you are using the Grape framework, you must configure your application to use the Ruby agent. A simple application configured to work with the Ruby agent looks like the following example:

```
require 'grape'
require 'contrast-agent'

class App < Grape::API
  use Contrast::Agent::Middleware, true
end
```

Configure Grape with config.ru

If there is no class extending, `Grape::API` or `config.ru` is the default way of racking up your application with the following configuration:

`config.ru`

```
# frozen_string_literal: true

require 'rack'
# Require Grape early for Framework support to detect it.
require 'Grape'

# example app.rb, could be any file implementing Grape
# endpoints and logic.
#
# For instance:
#
# # frozen_string_literal: true
#
# require 'Grape'
#
# class App
#   def initialize
#     @filenames = ['', '.html', 'index.html', '/index.html']
#     @rack_static = ::Rack::Static.new(
#       lambda { [404, {}, []] },
#       root: File.expand_path('../public', __dir__),
#       urls: ['/' ]
#     )
#   end
#
#   def self.instance
#     @instance ||= Rack::Builder.new do
#       use Rack::Cors do
#         allow do
#           origins '*'
#           resource '*', headers: :any, methods: :get
#         end
#       end
#
#       run App.new
#     end.to_app
#   end
#
#   def call(env)
#     # Grape::API impleted in API module:
#     API.call(env)
#     # handle response
#     .....
#   end
# end
#
require './app.rb'

# Contrast Agent needs to be required after Grape.
require 'contrast-agent'
```

```
# Example for requiring gems:
require 'bundler/setup'
Bundler.require(:default)

# Add Contrast Agent middleware to the rack stack:
use Contrast::Agent::Middleware, true

# Run Grape application:
run App.instance
```

Start the application with:

```
bundle exec rackup
```

OR

```
bundle exec rackup config.ru
```

See also

- [Configure the Ruby agent \(page 473\)](#)
- [Ruby supported technologies \(page 469\)](#)

Ruby web servers

Web Servers are technologies that deploy Web Application Frameworks. These servers manage the application processes and receive HTTP requests which they forward to said Web Application Frameworks.

Follow the guidelines based on your web server:

- [Passenger \(page 498\)](#)
- [Puma \(page 502\)](#)
- [Thin \(page 504\)](#)
- [Unicorn \(page 505\)](#)

See also

- [Configure the Ruby agent \(page 473\)](#)
- [Ruby supported technologies \(page 469\)](#)

Configure Passenger

Sometimes the Ruby agent pushes the application over the timeout threshold and that prevents the server from startup. This can be prevented by server configuration.

Passenger can be configured in Standalone mode or along side HTTP servers: *Passenger + NGINX* or *Passenger + Apache*.

The standard way of configuration in Standalone mode goes through these three options:

1. Command line options:

```
$ passenger start --start-timeout 100
```

2. Environment variables:

```
$ env PASSENGER_START_TIMEOUT=100 passenger start
```

3. Passengerfile.json (must be located in the application directory):

```
{  
  "start_timeout": "100"  
}
```

The order of configurations (most to least precedent):

- Command line options
- Environment variables
- Passengerfile.json



NOTE

An exception is made for **mass deployment**, then pre-app configurations are overridden both for command line and environment variables.

Passenger in NGINX or Apache mode is configured via the corresponding Apache or NGINX configuration files. These modes do not consult with Passengerfile.json.

```
# example of an Nginx configuration file which also configures Passenger:  
  
server {  
    server_name yourserver.com;  
    root /var/www/myapp/code/public;  
    passenger_enabled on;  
    passenger_ruby /usr/bin/ruby2.0;  
    passenger_sticky_sessions on;  
}
```

```
# example of an Apache configuration file which also configures Passenger:  
  
<VirtualHost *:80>  
    ServerName yourserver.com  
    DocumentRoot /var/www/myapp/code/public  
    PassengerStickySessions on  
  
    <Directory /var/www/myapp/code/public>  
        Allow from all  
        Options -MultiViews  
        Require all granted  
    </Directory>  
</VirtualHost>
```

Timeouts

- **Max request time** - The maximum amount of time, in seconds, that an application process may take to process a request. If the request takes longer than this amount of time, then the application process will be forcefully shut down, and possibly restarted upon the next request. A value of 0 means that there is no time limit. This is an enterprise only configuration.
 - Default value: 0
 - Command line:

```
$ passenger start --max-request-time SECONDS
```

- Environment variables:

- Passengerfile.json:

```
{
  "max_request_time": integer
}
```

- **Max request queue time** - When all concurrent requests are handled and their maximum number is reached, Passenger will queue all incoming requests. This option specifies the maximum time a request may spend in that queue. If a request in the queue reaches this specified limit, then Passenger will send a "504 Gateway Timeout" error for that request. A value of 0 means that the queue time is unbounded. This is an enterprise only configuration.

- Default value: 0

- Command line:

```
$ passenger start --max-request-queue-time NUMBER
```

- Environment variables:

```
PASSENGER_MAX_REQUEST_QUEUE_TIME=integer
```

- Passengerfile.json:

```
{
  "max_request_queue_time": integer
}
```

- **Pool idle time** - Maximum time for idle application process. If an application process hasn't received any traffic after the given number of seconds, then it will be shutdown in order to conserve memory. When this value is set to 0, application processes will not be shutdown unless manual killed or crush occurs. Decreasing this value means that application processes will have to be spawned more often.

- Default value: 300 (5 minutes)

- Command line:

```
$ passenger start --pool-idle-time SECONDS
```

- Environment variables:

```
PASSENGER_POOL_IDLE_TIME=integer
```

- Passengerfile.json:

```
{
  "pool_idle_time": integer
}
```

- **Max preload idle time** - Timeout for automatically shutdown a preloader process if it hasn't done anything for a given period. This option allows you to set the preloader's idle timeout, in seconds. A value of 0 means that it should never idle timeout. Setting a higher value will mean that the preloader is kept around longer, which may slightly increase memory usage. But as long as the preloader server is running, the time to spawn a Ruby application process only takes about 10% of the time that is normally needed.

- Default value: 300 (5 minutes)

- Command line:

```
$ passenger start --max-preloader-idle-time SECONDS
```

- Environment variables:

```
PASSENGER_MAX_PRELOADER_IDLE_TIME=integer
```

- Passengerfile.json:


```
{
  "max_preloader_idle_time": integer
}
```

- **Start timeout** - Timeout for the startup of application. If an application process fails to start within the timeout period then it will be forcefully killed with SIGKILL, and the error will be logged.

- Default value: 90
- Command line:

```
$ passenger start --start-timeout SECONDS;
```

- Environment variables:

```
PASSENGER_START_TIMEOUT=integer
```

- Passengerfile.json:

```
{
  "start_timeout": integer
}
```

Forking

Passenger is more like a process manager and instead of running application inside its own process space it launches it as external process and handle the management. This includes shut down of unused processes, or restarting them when they crash. An instance of an application is called a process. Passenger takes care of starting and stopping application.

Spawning is when Passenger starts an instance of an application. There are two methods for spawning an application in Passenger:

- **Direct:** new Ruby process with full copy of the application code and the web framework in memory. This approach uses more memory and takes more time to start.
- **Smart:** For Ruby apps this method is default. It begins with 'preloader' process. This process loads the entire application along with the web framework, by loading the file config.ru. The preloader process doesn't participate in request handling. Whenever new application process is needed the preloader spawns a child process (with the fork() system call).

The command used for creating a new fork is:

```
$ bundle exec passenger start --min-instances 2
```

This tells Passenger to keep 2 instances of the application.

- Default value: 1
- Default pool size for instances: 6
- Passengerfile.json:

```
{
  "max_pool_size": 6
}
```

When a request is handled, Passenger will pass it to the process with the least number of requests. If a process is killed, it is restarted automatically. Processes are also dynamically scaled, depending on traffic, spawning new forks up to the maximum pool number.

See also

- [Configure the Ruby agent \(page 473\)](#)
- [Ruby supported technologies \(page 469\)](#)

Configure Puma

Sometimes the Ruby agent pushes the application over the timeout threshold and that prevents the server from startup. This can be prevented by server configuration.

Puma can be configured directly through the CLI, in the `config/puma.rb` or `config.ru` files.

Timeouts

While the agent should work with the default or your custom configuration, it adds overhead to the first request. As such, you may need to increase timeouts, and here's how:



IMPORTANT

Some of the options are available only in Cluster Mode. All of the available options for the timeouts are listed in the [puma/dsl.rb](#).

- **persistent_timeout(seconds)** - Define how long persistent connections can be idle before Puma closes them. The seconds are passed as integers.
- **first_data_timeout(seconds)** - Define how long the tcp socket stays open, if no data has been received. The seconds are passed as integers.
- ***force_shutdown_after(val=:forever) *** - How long to wait for threads to stop when shutting them down. You can pass seconds too, but you can pass symbols between `:forever` and `:immediately`.
 - `:forever` - the value is set to -1
 - `:immediately` - the value is set to 0
 - `seconds` - it sets them directly as the timeout



NOTE

Puma always waits a couple of seconds before shutdown, even in immediately mode.

The following options are only available in cluster mode:

- **worker_timeout(seconds)** - Verifies that all workers have checked in to the master process within the given timeout. This timeout is to protect against dead or hung processes. Setting this value will not protect against slow requests. The minimum value is 6 seconds, the default value is 60 seconds.
- **worker_boot_timeout(seconds)** - change the default worker timeout for booting. If unspecified - it will default to the value of `worker_timeout`.
- **worker_shutdown_timeout(seconds)** - Set the timeout for worker shutdown.
- **wait_for_less_busy_worker(val=0.005)** - attempts to route traffic to less-busy workers by causing them to delay listening on the socket, allowing workers which are not processing any requests to pick up new requests first.



NOTE

This setting only works with MRI. For all other interpreters, this setting does nothing.

Puma initially sets two default timeout values:

- **DefaultWorkerTimeout** = 60
- **DefaultWorkerShutdownTimeout** = 30

To apply all of the timeouts settings, Puma must be configured to work in Cluster Mode.

Forking

Cluster mode is introduced in Puma 5, which allows Puma to fork workers from worker 0, instead of directly from the master process.

Similar to the `preload_app` option, the `fork_worker` option allows your application to be initialized only once for copy-on-write memory savings.

This actual mode has couple of advantages, and the first one is that it's compatible with a phased restart. The master process initially does not preload the application and that's why this mode works with phased restart. When worker 0 reloads as part of a phased restart, it initializes a new copy of your application first, then the other workers reload by forking from this new worker already containing the new preloaded application.



TIP

A phased restart replaces all running workers in Puma cluster. It is done by first killing an old worker, then starting a new worker, waiting until the new worker has successfully started before proceeding to the next worker, until it goes through all workers. The master process is not restarted.

This allows a phased restart to complete as quickly as a hot restart while still minimizing downtime by staggering the restart across cluster workers.

The other advantage is that a `refork` command is added for additional copy-on-write improvements in running applications and the idea is that it re-loads all nonzero workers by re-forking them from worker 0.

This command can potentially improve memory utilization in large or complex applications that don't fully pre-initialize on startup, because the re-forked workers can share copy-on-write memory with a worker that has been running for a while and serving requests.

A refork will also automatically trigger once, after a certain number of requests have been processed by worker 0 (default 1000). To configure the number of requests before the auto-refork, pass a positive integer argument to `fork_worker` (e.g., `fork_worker 1000`), or 0 to disable.

Limitations:

- Cluster mode is not compatible with `preload_app`.
- In order to fork new workers cleanly, worker 0 shuts down its server and stops serving requests so there are no open file descriptors or other kinds of shared global state between processes, and to maximize copy-on-write efficiency across the newly-forked workers. This may temporarily reduce total capacity of the cluster during a phased restart / refork.

After going through `fork_worker` and `re-fork` commands, these are other clustered (fork worker) commands:

- ***workers(count)** *- How many worker processes to run. Typically this is set to the number of available cores. The default is the value of the environment variable `WEB_CONCURRENCY` if set, otherwise 0.

- **before_fork(&block)** - code to run immediately before master process forks workers (once on boot). These hooks can block if necessary to wait for background operations unknown to Puma to finish before the process terminates.
- **on_worker_boot(&block)** - code to run in a worker when it boots to setup the process before booting the app.
- **on_worker_shutdown(&block)** - code to run immediately before a worker shuts down (after it has finished processing HTTP requests)
- **on_worker_fork(&block)** - code to run in the master right before a worker is started. The worker's index is passed as an argument.
- **after_worker_fork(&block)** - code to run in the master after a worker has been started. The worker's index is passed as an argument.
- **on_refork(&block)** - When enabled, code to run in Worker 0 before all other workers are re-forked from this process, after the server has temporarily stopped serving requests (once per complete refork cycle). This can be used to trigger extra garbage-collection to maximize copy-on-write efficiency, or close any connections to remote servers(database, Redis, ...) that were opened while the server was running.
- **out_of_band(&block)** - code to run out-of-band when the worker is idle. These hooks run immediately after a request has finished processing and there are no busy threads on the worker. The worker doesn't accept new requests until this code finishes. This hook is useful for running out-of-band garbage collection or scheduling asynchronous tasks to execute after a response.
- **fork_worker(after_request=1000)** - When enabled, workers will be forked from worker 0 instead of from the master process. This option is similar to `preload_app` because the app is preloaded before forking, but it is compatible with phased restart. This option also enables the `refork` command.
- **nakayoshi_fork(enabled=true)** - This is kind of different, but when enabled, Puma will GC 4 times before forking workers. It will increase time to boot and fork. See your logs for details on how much time this adds to your boot process. For most apps, it will be less than one second. This fork method is based on the work of Koichi Sasada and Aaron Patterson and this option may decrease memory utilization of preload-enabled cluster-mode Pumas.



NOTE

If available (Ruby 2.7+), it will also call `GC.compact`.

Not recommended for non-MRI Rubies.

See also

- [Configure the Ruby agent \(page 473\)](#)
- [Ruby supported technologies \(page 469\)](#)

Configure Thin

Sometimes the Ruby agent pushes the application over the timeout threshold and that prevents the server from startup. This can be prevented by server configuration.

Thin is lightweight web server, supporting clusters and providing options to set timeouts and wait time. Thin accepts CLI commands or adding a `config.yml` file in which you can set all the settings you need.

Timeouts

The **wait** option is the maximum wait time for the server to be restarted within a cluster.

The **timeout** option is the maximum number of seconds for incoming data to arrive before the connection is dropped.

Forking

Thin supports clusters and you can run several server instances on different ports.

These are the available options:

```
cluster options:
-s, --servers NUM           Number of servers to start
-o, --only NUM              Send command to only one server of the
cluster
-C, --config FILE           Load options from config file
-O, --onebyone              Restart the cluster one by one (only works
with restart command)
-w, --wait NUM              Maximum wait time for server to be started
in seconds (use with -O)
```

There is also an experimental tuning option that can be run alongside clusters:

```
--threaded                  Call the Rack application in threads
[experimental]
```

See also

- [Configure the Ruby agent \(page 473\)](#)
- [Ruby supported technologies \(page 469\)](#)

Configure Unicorn

Unicorn is single-threaded and multi-processed. It doesn't adjust the number of processes automatically based on traffic. You must use a `unicorrf.conf.rb` or `unicorn.conf.minimal.rb` file, or a script, to configure this.

- Example for `unicorn.conf.rb`:

```
# Define your root directory.
root = "/home/deployer/apps/gifroll/current"

# Define worker directory for Unicorn.
working_directory "/path/to/app/current"

# Define number of worker processes.
# Each forked OS process consumes additional memory.
worker_processes 4

# Define timeout for hanging workers before they are restarted.
timeout 30

# Location of PID file.
pid "/path/to/app/shared/pids/unicorn.pid"

# Define log paths:

# Allow redirecting $stderr to a given path. Unlike doing this from the
shell,
# this allows the Unicorn process to know the path being written to and
rotates
# the file if it is used for logging.
stderr_path "#{root}/log/unicorn.log"
```

```
# Same as stderr_path, except for $stdout. Not many Rack applications
write
# to $stdout, but any that do will have their output written here.
stdout_path "#{root}/log/unicorn.log"

# Loads Rails before forking workers for better worker spawn time.
# Preloading your application reduces the startup time of individual
# Unicorn worker_processes and allows you to manage the external
connections
# of each individual worker using the before_fork and after_fork calls.
#
# Please check if other external connections work properly with
# Unicorn forking. Many popular gems (dalli memcache client, Redis) will
have
# compatibility confirmation with Unicorn and the process model.
# Check the gem documentation for more information.
preload_app true

# When enabled, Unicorn will check the client connection by writing the
# beginning of the HTTP headers before calling the application. This will
# prevent calling the application for clients who have disconnected while
# their connection was queued.
check_client_connection false

# Enable a local variable to guard against running a hook (before_fork,
after_fork)
# multiple times
run_once = true

# For example, use of before_fork and after_fork:
#
# POSIX Signals are a form of interprocess communication, and signal
# events or state changes.
# QUIT: Signals a process to exit, but creates a core dump.
# TERM: Tells a process to terminate, but allows the process
# to clean up after itself.
#
# Unicorn uses the QUIT signal to indicate a graceful shutdown.
# The master process receives it and sends it to all workers, telling
them to
# shutdown after any in-flight request.
before_fork do |server, worker|
  Signal.trap 'TERM' do
    puts 'Unicorn master intercepting TERM and sending myself QUIT
instead'
    Process.kill 'QUIT', Process.pid
  end

  # You may want to execute code in the master process, before the forking
  # begins, to deal with operations that causes changes in state.
  # You need them to run once:
  if run_once
    # do_something_once_here ...
    run_once = false # prevent from firing again
  end
end
```

```

end

after_fork do |server, worker|
  Signal.trap 'TERM' do
    puts 'Unicorn worker intercepting TERM and doing nothing. Wait for
master to send QUIT'
  end
  # ...
end

# For more information, check the Unicorn Configurator: https://msp-
greg.github.io/unicorn/Unicorn/Configurator.html

```

- Example for `unicorn.conf.minimal.rb`:

```

listen 2007 # by default Unicorn listens on port 8080
worker_processes 2 # this should be >= nr_cpus
pid "/path/to/app/shared/pids/unicorn.pid"
stderr_path "/path/to/app/shared/log/unicorn.log"
stdout_path "/path/to/app/shared/log/unicorn.log"

```

Configure forking

Unicorn has a multi-process architecture to make better use of available CPU cores. On startup, the Unicorn master process loads the application code and then spawns workers which inherit the application code from their master process. The requests are handled only by the workers and never by the master. The operating system network stack queues incoming requests and distributes them among the workers.

Unicorn is designed to replace crashed workers without dropping user requests. When a worker reaches the request timeout, the master process ends it (with `kill -9`) and replaces it with a new process. You can configure the number of worker processes and the request timeout.

Configurations	Description
<code>worker_processes (nr)</code>	<p>Sets the current number of <code>worker_processes</code> to <code>nr</code>. Each worker process will serve exactly one client at a time. You can increment or decrement this value at runtime by sending signals <code>SIGTTIN</code> or <code>SIGTTOU</code> respectively to the master process without reloading the rest of your Unicorn configuration.</p> <p>You can read more about signals on Unicorn's site.</p>
<code>Unicorn::Configurator#after_fork</code>	Sets <code>after_fork</code> hook to a given block
<code>Unicorn::Configurator#before_fork</code>	Sets <code>before_fork</code> hook to a given block.
<code>Unicorn::Configurator#preload_app(bool)</code> <code>ObjectEnabling</code>	This preloads an application before forking worker processes. This allows memory savings when using a copy-on-write-friendly GC but can cause bad things to happen when resources like sockets are opened at load time by the master process and shared by multiple children.

Configure timeouts

Variable	Description	Default value
<code>timeout 30</code>	This sets the timeout of worker processes to 30 seconds, the default value is 60 seconds. The timeout configuration will end workers that exceed this limit. This timeout is enforced by the master process itself and not subject to the scheduling limitations by the worker process.	60 seconds
<code>delay integer</code>	This sets the seconds to wait between successful tries.	0.5 seconds

**TIP**

Unicorn works with [nginx](#) for more settings.

Configure APMs

The following APMs support Unicorn:

- [Scout](#) has a [separate class for Unicorn integration](#).
- [New Relic](#)
- [AppDynamics](#)

See also

- [Configure the Ruby agent \(page 473\)](#)
- [Ruby supported technologies \(page 469\)](#)

Ruby YAML template

Use this template to configure the Ruby agent using a YAML configuration file. (Learn more about [YAML configuration \(page 107\)](#).)

Place your YAML file in the default location: `/etc/contrast/contrast_security.yaml`

```
#
=====
==
# Use the properties in this YAML file to configure a Contrast agent.
# Go to https://docs.contrastsecurity.com/en/order-of-precedence.html
# to determine the order of precedence for configuration values.
#
=====
==

# Use this setting if you want to temporarily disable a Contrast agent.
# Set to `true` to enable the agent; set to `false` to disable the agent.
# enable: true

#
=====
==
# api
# Use the properties in this section to connect the agent to the Contrast
UI.
#
=====
==
api:

  # ***** REQUIRED *****
  # Set the URL for the Contrast UI.
  url: https://app.contrastsecurity.com/Contrast
```



```
# ***** REQUIRED *****
# Set the API key needed to communicate with the Contrast UI.
api_key: NEEDS_TO_BE_SET

# ***** REQUIRED *****
# Set the service key needed to communicate with the Contrast
# UI. It is used to calculate the Authorization header.
service_key: NEEDS_TO_BE_SET

# ***** REQUIRED *****
# Set the user name used to communicate with the Contrast
# UI. It is used to calculate the Authorization header.
user_name: NEEDS_TO_BE_SET

#
=====
# api.certificate
# Use the following properties for communication
# with the Contrast UI using certificates.
#
=====
# certificate:

# If set to `false`, the agent will ignore the
# certificate configuration in this section.
# enable: true

# Set the absolute or relative path to a CA for communication
# with the Contrast UI using a self-signed certificate.
# ca_file: NEEDS_TO_BE_SET

# Set the absolute or relative path to the Certificate
# PEM file for communication with the Contrast UI.
# cert_file: NEEDS_TO_BE_SET

# Set the absolute or relative path to the Key PEM
# file for communication with the Contrast UI.
# key_file: NEEDS_TO_BE_SET

#
=====
# api.proxy
# Use the following properties for communication
# with the Contrast UI over a proxy.
#
=====
# proxy:

# Set value to `true` for the agent to communicate with
# the Contrast web interface over a proxy. Set value to
# `false` if you don't want to use the proxy. If no value is
# indicated, the presence of a valid **contrast.proxy.host**
# and **contrast.proxy.port** will enable the proxy.
# enable: NEEDS_TO_BE_SET
```

```
# Set the URL for your Proxy Server. The URL form is `scheme://
host:port`.
# url: NEEDS_TO_BE_SET

#
=====
==
# agent
# Use the properties in this section to control the way and frequency
# with which the agent communicates to logs and the Contrast UI.
#
=====
==
# agent:

#
=====
# agent.logger
# Define the following properties to set logging values.
# If the following properties are not defined, the
# agent uses the logging values from the Contrast UI.
#
=====
# logger:

# Enable diagnostic logging by setting a path to a log file.
# While diagnostic logging hurts performance, it generates
# useful information for debugging Contrast. The value set here
# is the location to which the agent saves log output. If no
# log file exists at this location, the agent creates a file.
#
# Example - `/opt/Contrast/contrast.log` creates a log in the
# `/opt/Contrast` directory, and rotates it automatically as needed.
#
# path: ./contrast_agent.log

# Set the the log output level. Valid options are
# `ERROR`, `WARN`, `INFO`, `DEBUG`, and `TRACE`.
# level: INFO

# Override the name of the process the agents uses in logs.
# progname: Contrast Agent

# Set to `true` for the agent to tag
# logs with `!AM!` for the metrics tool.
# metrics: true

#
=====
# agent.security_logger
# Define the following properties to set security logging
# values associated with Protect. If not defined, the agent
# uses the security logging (CEF) values from the Contrast UI.
#
=====
```

```
# security_logger:

# Set the file to which the agent logs security events.
# path: ./contrast/security.log

# Set the log level for security logging. Valid options
# are `ERROR`, `WARN`, `INFO`, `DEBUG`, and `TRACE`.
# level: ERROR

#
=====
# agent.security_logger.syslog
# Define the following properties to set Syslog values. If the
properties
# are not defined, the agent uses the Syslog values from the Contrast
UI.
#
=====
# syslog:

# Set to `true` to enable Syslog logging.
# enable: NEEDS_TO_BE_SET

# Set the IP address of the Syslog server
# to which the agent should send messages.
# ip: NEEDS_TO_BE_SET

# Set the port of the Syslog server to
# which the agent should send messages.
# port: NEEDS_TO_BE_SET

# Set the facility code of the messages the agent sends to Syslog.
# facility: 19

# Set the log level of Exploited attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_exploited: ALERT

# Set the log level of Blocked attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_blocked: NOTICE

# Set the log level of Blocked At Perimeter
# attacks. Value options are `ALERT`, `CRITICAL`,
# `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_blocked_perimeter: NOTICE

# Set the log level of Probed attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_probed: WARNING

# Set the log level of Suspicious attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_suspicious: WARNING
```

```
#
=====
# agent.heap_dump
# The following properties are used to trigger heap dumps from within
# the agent to snapshot the behavior of instrumented applications.
#
=====
# heap_dump:

# Set to `true` for the agent to automatically
# take heap dumps of the instrumented application.
# enable: false

# Set the location to which to save the heap dump files. If relative,
# the path is determined based on the process' working directory.
# path: contrast_heap_dumps

# Set the amount of time to wait, in milliseconds,
# after agent startup to begin taking heap dumps.
# delay_ms: 10_000

# Set the amount of time to wait, in milliseconds, between each heap
dump.
# window_ms: 10_000

# Set the number of heap dumps to take before disabling this feature.
# count: 5

# Set to `true` for the agent to trigger garbage collection before
# taking a heap dump to remove temporary objects from the dump.
# clean: false

#
=====
# agent.ruby
# The following properties apply to any Ruby agent-wide configurations.
#
=====
# ruby:

# Allow the agent to track frozen Objects returned by
# source methods. This configuration is on by default.
# track_frozen_sources: NEEDS_TO_BE_SET

# Allow the agent to track propagation through interpolated
# Strings. This configuration is on by default.
# interpolate: NEEDS_TO_BE_SET

# Set a comma-separated string of rake tasks
# in which to disable agent operation.
# disabled_agent_rake_tasks:
about,assets:clean,assets:clobber,assets:environment,assets:precompile,asset
s:precompile:all,db:create,db:drop,db:migrate:status,db:rollback,db:schema:c
ache:clear,db:schema:cache:dump,db:schema:dump,db:schema:load,db:seed,db:set
up,db:structure:dump,db:version,doc:app,log:clear,middleware,notes,notes:cus
```

```
tom,rails:template,rails:update,routes,secret,spec,spec:features,spec:requests,spec:controllers,spec:helpers,spec:models,spec:views,spec:routing,spec:roov,stats,test,test:all,test:all:db,test:recent,test:single,test:uncommitted,time:zones:all,tmp:clear,tmp:create,webpacker:compile

#
=====
==
# inventory
# Use the properties in this section to override the inventory features.
#
=====
==
# inventory:

# Set to `false` to disable inventory features in the agent.
# enable: true

# Apply a list of labels to libraries. Labels
# must be formatted as a comma-delimited list.
# Example - `label1, label2, label3`
#
# tags: NEEDS_TO_BE_SET

#
=====
==
# assess
# Use the properties in this section to control Assess.
#
=====
==
# assess:

# Include this property to determine if the Assess
# feature should be enabled. If this property is not
# present, the decision is delegated to the Contrast UI.
# enable: false

# Apply a list of labels to vulnerabilities and preflight
# messages. Labels must be formatted as a comma-delimited list.
# Example - `label1, label2, label3`
#
# tags: NEEDS_TO_BE_SET

# Value options are `ALL`, `SOME`, or `NONE`.
# stacktraces: ALL

#
=====
# assess.sampling
# Use the following properties to control sampling in the agent.
#
=====
# sampling:
```

```
# Set to `true` to enable sampling.
# enable: false

# This property indicates the number of requests
# to analyze in each window before sampling begins.
# baseline: 5

# This property indicates that every *nth*
# request after the baseline is analyzed.
# request_frequency: 10

# This property indicates the duration for which a sample set is valid.
# window_ms: 180_000

#
=====
# assess.rules
# Use the following properties to control simple rule configurations.
#
=====
# rules:

# Define a list of Assess rules to disable in the agent. To view a
# list of rule names, in Contrast go to user menu > Policy Management >
# Assess rules. The rules must be formatted as a comma-delimited list.
#
# Example - Set `reflected-xss,sql-injection` to disable
# the reflected-xss rule and the sql-injection rule.
#
# disabled_rules: NEEDS_TO_BE_SET

#
=====
==
# protect
# Use the properties in this section to override Protect features.
#
=====
==
# protect:

# Include this property to determine if the Protect
# feature should be enabled. If this property is not
# present, the decision is delegated to the Contrast UI.
# enable: false

#
=====
# protect.rules
# Use the following properties to set simple rule configurations.
#
=====
# rules:
```

```
# Define a list of Protect rules to disable in the agent. To view a
# list of rule names, in Contrast go to user menu > Policy Management >
# Protect rules. The rules must be formatted as a comma-delimited list.
# disabled_rules: NEEDS_TO_BE_SET

#
=====
# protect.rules.bot-blocker
# Use the following selection to configure if the
# agent blocks bots. Set to `true` to enable blocking.
#
=====
# bot-blocker:

# Set to `true` for the agent to block known bots.
# enable: false

#
=====
# protect.rules.sql-injection
# Use the following settings to configure the sql-injection rule.
#
=====
# sql-injection:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or off.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.cmd-injection
# Use the following properties to configure
# how the command injection rule works.
#
=====
# cmd-injection:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.path-traversal
# Use the following properties to configure
# how the path traversal rule works.
```

```
#
=====
# path-traversal:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.method-tampering
# Use the following properties to configure
# how the method tampering rule works.
#
=====
# method-tampering:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.reflected-xss
# Use the following properties to configure how
# the reflected cross-site scripting rule works.
#
=====
# reflected-xss:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.xxe
# Use the following properties to configure
# how the XML external entity works.
#
=====
# xxe:
```



```
# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
==
# application
# Use the properties in this section for
# the application(s) hosting this agent.
#
=====
==
# application:

# Override the reported application name.
#
# Note - On systems where multiple, distinct applications may be served
# by a single process, this configuration causes the agent to report
# all discovered applications as one application with the given name.
#
# Note - On Java systems where multiple, distinct applications may be
# served by a single process, this configuration causes the agent to
report
# all discovered applications as one application with the given name.
#
# name: NEEDS_TO_BE_SET

# Override the reported application path.
# path: NEEDS_TO_BE_SET

# Add the name of the application group with which this
# application should be associated in the Contrast UI.
# group: NEEDS_TO_BE_SET

# Add the application code this application should use in the Contrast UI.
# code: NEEDS_TO_BE_SET

# Override the reported application version.
# version: NEEDS_TO_BE_SET

# Apply labels to an application. Labels must
# be formatted as a comma-delimited list.
# Example - `label1,label2,label3`
#
# tags: NEEDS_TO_BE_SET

# Define a set of `key=value` pairs (which conforms to RFC 2253) for
# specifying user-defined metadata associated with the application. The
# set must be formatted as a comma-delimited list of `key=value` pairs.
# Example - `business-unit=accounting, office=Baltimore`
```

```
#
# metadata: NEEDS_TO_BE_SET

# Provide the ID of a session which already exists in the Contrast
# UI. Vulnerabilities discovered by the agent are associated with
# this session. If an invalid ID is supplied, the agent will be
# disabled. This option and `application.session_metadata` are
# mutually exclusive; if both are set, the agent will be disabled.
# session_id: NEEDS_TO_BE_SET

# Provide metadata which is used to create a new session ID in the
# Contrast UI. Vulnerabilities discovered by the agent are associated with
# this new session. This value should be formatted as `key=value` pairs
# (conforming to RFC 2253). Available key names for this configuration
# are branchName, buildNumber, commitHash, committer, gitTag, repository,
# testRun, and version. This option and `application.session_id` are
# mutually exclusive; if both are set the agent will be disabled.
# session_metadata: NEEDS_TO_BE_SET

#
=====
==
# server
# Use the settings in this section to set metadata for the server
# hosting this agent. Contrast recognizes common, supported server
# names, paths, types and environments. Doing this may require a new
# server or license, and it may affect functionality of some features.
#
=====
==
# server:

# Override the reported server name.
# name: localhost

# Override the reported server path.
# path: NEEDS_TO_BE_SET

# Override the reported server type.
# type: NEEDS_TO_BE_SET

# Set the environment directly to override the default set
# by the Contrast UI. This allows the user to configure the
# environment dynamically at startup rather than manually
# updating the Server in the Contrast UI themselves afterwards.
#
# Valid values include `QA`, `PRODUCTION` and `DEVELOPMENT`.
# For example, `PRODUCTION` registers this Server as
# running in a `PRODUCTION` environment, regardless of the
# organization's default environment in the Contrast UI.
#
# environment: NEEDS_TO_BE_SET

# Apply a list of labels to the server. Labels
# must be formatted as a comma-delimited list.
```

```
# Example - `label1,label2,label3`
#
# tags: NEEDS_TO_BE_SET

#
=====
==
# Use the properties in this YAML file to configure a Contrast agent.
# Go to https://docs.contrastsecurity.com/en/order-of-precedence.html
# to determine the order of precedence for configuration values.
#
=====
==

# Use this setting if you want to temporarily disable a Contrast agent.
# Set to `true` to enable the agent; set to `false` to disable the agent.
# enable: true

#
=====
==
# api
# Use the properties in this section to connect the agent to the Contrast
# UI.
#
=====
==
api:

# ***** REQUIRED *****
# Set the URL for the Contrast UI.
url: https://app.contrastsecurity.com/Contrast

# ***** REQUIRED *****
# Set the API key needed to communicate with the Contrast UI.
api_key: NEEDS_TO_BE_SET

# ***** REQUIRED *****
# Set the service key needed to communicate with the Contrast
# UI. It is used to calculate the Authorization header.
service_key: NEEDS_TO_BE_SET

# ***** REQUIRED *****
# Set the user name used to communicate with the Contrast
# UI. It is used to calculate the Authorization header.
user_name: NEEDS_TO_BE_SET

#
=====
# api.certificate
# Use the following properties for communication
# with the Contrast UI using certificates.
#
=====
```

```
# certificate:

# If set to `false`, the agent will ignore the
# certificate configuration in this section.
# enable: true

# Set the absolute or relative path to a CA for communication
# with the Contrast UI using a self-signed certificate.
# ca_file: NEEDS_TO_BE_SET

# Set the absolute or relative path to the Certificate
# PEM file for communication with the Contrast UI.
# cert_file: NEEDS_TO_BE_SET

# Set the absolute or relative path to the Key PEM
# file for communication with the Contrast UI.
# key_file: NEEDS_TO_BE_SET

#
=====
# api.proxy
# Use the following properties for communication
# with the Contrast UI over a proxy.
#
=====
# proxy:

# Set value to `true` for the agent to communicate with
# the Contrast web interface over a proxy. Set value to
# `false` if you don't want to use the proxy. If no value is
# indicated, the presence of a valid **contrast.proxy.host**
# and **contrast.proxy.port** will enable the proxy.
# enable: NEEDS_TO_BE_SET

# Set the URL for your Proxy Server. The URL form is `scheme://
host:port`.
# url: NEEDS_TO_BE_SET

#
=====
==
# agent
# Use the properties in this section to control the way and frequency
# with which the agent communicates to logs and the Contrast UI.
#
=====
==
# agent:

#
=====
# agent.logger
# Define the following properties to set logging values.
# If the following properties are not defined, the
# agent uses the logging values from the Contrast UI.
```

```
#
=====
# logger:

# Enable diagnostic logging by setting a path to a log file.
# While diagnostic logging hurts performance, it generates
# useful information for debugging Contrast. The value set here
# is the location to which the agent saves log output. If no
# log file exists at this location, the agent creates a file.
#
# Example - `/opt/Contrast/contrast.log` creates a log in the
# `/opt/Contrast` directory, and rotates it automatically as needed.
#
# path: ./contrast_agent.log

# Set the the log output level. Valid options are
# `ERROR`, `WARN`, `INFO`, `DEBUG`, and `TRACE`.
# level: INFO

# Override the name of the process the agents uses in logs.
# progname: Contrast Agent

# Set to `true` for the agent to tag
# logs with `!AM!` for the metrics tool.
# metrics: true

#
=====
# agent.security_logger
# Define the following properties to set security logging
# values associated with Protect. If not defined, the agent
# uses the security logging (CEF) values from the Contrast UI.
#
=====
# security_logger:

# Set the file to which the agent logs security events.
# path: ./contrast/security.log

# Set the log level for security logging. Valid options
# are `ERROR`, `WARN`, `INFO`, `DEBUG`, and `TRACE`.
# level: ERROR

#
=====
# agent.security_logger.syslog
# Define the following properties to set Syslog values. If the
properties
# are not defined, the agent uses the Syslog values from the Contrast
UI.
#
=====
# syslog:

# Set to `true` to enable Syslog logging.
```

```
# enable: NEEDS_TO_BE_SET

# Set the IP address of the Syslog server
# to which the agent should send messages.
# ip: NEEDS_TO_BE_SET

# Set the port of the Syslog server to
# which the agent should send messages.
# port: NEEDS_TO_BE_SET

# Set the facility code of the messages the agent sends to Syslog.
# facility: 19

# Set the log level of Exploited attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_exploited: ALERT

# Set the log level of Blocked attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_blocked: NOTICE

# Set the log level of Blocked At Perimeter
# attacks. Value options are `ALERT`, `CRITICAL`,
# `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_blocked_perimeter: NOTICE

# Set the log level of Probed attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_probed: WARNING

# Set the log level of Suspicious attacks. Value options are `ALERT`,
# `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
# severity_suspicious: WARNING

#
=====
# agent.heap_dump
# The following properties are used to trigger heap dumps from within
# the agent to snapshot the behavior of instrumented applications.
#
=====
# heap_dump:

# Set to `true` for the agent to automatically
# take heap dumps of the instrumented application.
# enable: false

# Set the location to which to save the heap dump files. If relative,
# the path is determined based on the process' working directory.
# path: contrast_heap_dumps

# Set the amount of time to wait, in milliseconds,
# after agent startup to begin taking heap dumps.
# delay_ms: 10_000
```

```

# Set the amount of time to wait, in milliseconds, between each heap
dump.
# window_ms: 10_000

# Set the number of heap dumps to take before disabling this feature.
# count: 5

# Set to `true` for the agent to trigger garbage collection before
# taking a heap dump to remove temporary objects from the dump.
# clean: false

#
=====
# agent.ruby
# The following properties apply to any Ruby agent-wide configurations.
#
=====
# ruby:

# Allow the agent to track frozen Objects returned by
# source methods. This configuration is on by default.
# track_frozen_sources: NEEDS_TO_BE_SET

# Allow the agent to track propagation through interpolated
# Strings. This configuration is on by default.
# interpolate: NEEDS_TO_BE_SET

# Set a comma-separated string of rake tasks
# in which to disable agent operation.
# disabled_agent_rake_tasks:
about,assets:clean,assets:clobber,assets:environment,assets:precompile,asset
s:precompile:all,db:create,db:drop,db:migrate:status,db:rollback,db:schema:c
ache:clear,db:schema:cache:dump,db:schema:dump,db:schema:load,db:seed,db:set
up,db:structure:dump,db:version,doc:app,log:clear,middleware,notes,notes:cus
tom,rails:template,rails:update,routes,secret,spec,spec:features,spec:reques
ts,spec:controllers,spec:helpers,spec:models,spec:views,spec:routing,spec:rc
ov,stats,test,test:all,test:all:db,test:recent,test:single,test:uncommitted,
time:zones:all,tmp:clear,tmp:create,webpacker:compile

#
=====
==
# inventory
# Use the properties in this section to override the inventory features.
#
=====
==
# inventory:

# Set to `false` to disable inventory features in the agent.
# enable: true

# Apply a list of labels to libraries. Labels
# must be formatted as a comma-delimited list.
# Example - `label1, label2, label3`

```

```
#
# tags: NEEDS_TO_BE_SET

#
=====
==
# assess
# Use the properties in this section to control Assess.
#
=====
==
# assess:

# Include this property to determine if the Assess
# feature should be enabled. If this property is not
# present, the decision is delegated to the Contrast UI.
# enable: false

# Apply a list of labels to vulnerabilities and preflight
# messages. Labels must be formatted as a comma-delimited list.
# Example - `label1, label2, label3`
#
# tags: NEEDS_TO_BE_SET

# Value options are `ALL`, `SOME`, or `NONE`.
# stacktraces: ALL

#
=====
# assess.sampling
# Use the following properties to control sampling in the agent.
#
=====
# sampling:

# Set to `true` to enable sampling.
# enable: false

# This property indicates the number of requests
# to analyze in each window before sampling begins.
# baseline: 5

# This property indicates that every *nth*
# request after the baseline is analyzed.
# request_frequency: 10

# This property indicates the duration for which a sample set is valid.
# window_ms: 180_000

#
=====
# assess.rules
# Use the following properties to control simple rule configurations.
#
=====
```



```
# rules:

# Define a list of Assess rules to disable in the agent. To view a
# list of rule names, in Contrast go to user menu > Policy Management >
# Assess rules. The rules must be formatted as a comma-delimited list.
#
# Example - Set `reflected-xss,sql-injection` to disable
# the reflected-xss rule and the sql-injection rule.
#
# disabled_rules: NEEDS_TO_BE_SET

#
=====
==
# protect
# Use the properties in this section to override Protect features.
#
=====
==
# protect:

# Include this property to determine if the Protect
# feature should be enabled. If this property is not
# present, the decision is delegated to the Contrast UI.
# enable: false

#
=====
# protect.rules
# Use the following properties to set simple rule configurations.
#
=====
# rules:

# Define a list of Protect rules to disable in the agent. To view a
# list of rule names, in Contrast go to user menu > Policy Management >
# Protect rules. The rules must be formatted as a comma-delimited list.
# disabled_rules: NEEDS_TO_BE_SET

#
=====
# protect.rules.bot-blocker
# Use the following selection to configure if the
# agent blocks bots. Set to `true` to enable blocking.
#
=====
# bot-blocker:

# Set to `true` for the agent to block known bots.
# enable: false

#
=====
# protect.rules.sql-injection
# Use the following settings to configure the sql-injection rule.
```

```
#
=====
# sql-injection:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or off.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.cmd-injection
# Use the following properties to configure
# how the command injection rule works.
#
=====
# cmd-injection:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.path-traversal
# Use the following properties to configure
# how the path traversal rule works.
#
=====
# path-traversal:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.method-tampering
# Use the following properties to configure
# how the method tampering rule works.
#
=====
# method-tampering:
```

```
# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.reflected-xss
# Use the following properties to configure how
# the reflected cross-site scripting rule works.
#
=====
# reflected-xss:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
# protect.rules.xxe
# Use the following properties to configure
# how the XML external entity works.
#
=====
# xxe:

# Set the mode of the rule. Value options are
# `monitor`, `block`, `block_at_perimeter`, or `off`.
#
# Note - If a setting says, "if blocking is enabled",
# the setting can be `block` or `block_at_perimeter`.
#
# mode: off

#
=====
==
# application
# Use the properties in this section for
# the application(s) hosting this agent.
#
=====
==
# application:

# Override the reported application name.
#
```

```
# Note - On systems where multiple, distinct applications may be served
# by a single process, this configuration causes the agent to report
# all discovered applications as one application with the given name.
#
# Note - On Java systems where multiple, distinct applications may be
# served by a single process, this configuration causes the agent to
report
# all discovered applications as one application with the given name.
#
# name: NEEDS_TO_BE_SET

# Override the reported application path.
# path: NEEDS_TO_BE_SET

# Add the name of the application group with which this
# application should be associated in the Contrast UI.
# group: NEEDS_TO_BE_SET

# Add the application code this application should use in the Contrast UI.
# code: NEEDS_TO_BE_SET

# Override the reported application version.
# version: NEEDS_TO_BE_SET

# Apply labels to an application. Labels must
# be formatted as a comma-delimited list.
# Example - `label1,label2,label3`
#
# tags: NEEDS_TO_BE_SET

# Define a set of `key=value` pairs (which conforms to RFC 2253) for
# specifying user-defined metadata associated with the application. The
# set must be formatted as a comma-delimited list of `key=value` pairs.
# Example - `business-unit=accounting, office=Baltimore`
#
# metadata: NEEDS_TO_BE_SET

# Provide the ID of a session which already exists in the Contrast
# UI. Vulnerabilities discovered by the agent are associated with
# this session. If an invalid ID is supplied, the agent will be
# disabled. This option and `application.session_metadata` are
# mutually exclusive; if both are set, the agent will be disabled.
# session_id: NEEDS_TO_BE_SET

# Provide metadata which is used to create a new session ID in the
# Contrast UI. Vulnerabilities discovered by the agent are associated with
# this new session. This value should be formatted as `key=value` pairs
# (conforming to RFC 2253). Available key names for this configuration
# are branchName, buildNumber, commitHash, committer, gitTag, repository,
# testRun, and version. This option and `application.session_id` are
# mutually exclusive; if both are set the agent will be disabled.
# session_metadata: NEEDS_TO_BE_SET

#
=====
```

```
==
# server
# Use the settings in this section to set metadata for the server
# hosting this agent. Contrast recognizes common, supported server
# names, paths, types and environments. Doing this may require a new
# server or license, and it may affect functionality of some features.
#
=====
==
# server:

# Override the reported server name.
# name: localhost

# Override the reported server path.
# path: NEEDS_TO_BE_SET

# Override the reported server type.
# type: NEEDS_TO_BE_SET

# Set the environment directly to override the default set
# by the Contrast UI. This allows the user to configure the
# environment dynamically at startup rather than manually
# updating the Server in the Contrast UI themselves afterwards.
#
# Valid values include `QA`, `PRODUCTION` and `DEVELOPMENT`.
# For example, `PRODUCTION` registers this Server as
# running in a `PRODUCTION` environment, regardless of the
# organization's default environment in the Contrast UI.
#
# environment: NEEDS_TO_BE_SET

# Apply a list of labels to the server. Labels
# must be formatted as a comma-delimited list.
# Example - `label1,label2,label3`
#
# tags: NEEDS_TO_BE_SET
```

Ruby telemetry

The Ruby agent use telemetry to collect usage data. Telemetry is collected when an instrumented application first loads the agent's sensors and then periodically (every few hours) afterwards.

[Your privacy is important to us \(page 1275\)](#). The telemetry feature does not collect application data. The data is anonymized before being sent securely to Contrast. Then the aggregated data is stored encrypted and under restricted access control. Any collected data will be deleted after one year.

The telemetry feature collects the following data:

Agent versions	Data
Ruby 4.13	Agent version
	Operating system and version
	Ruby version
	Application framework and version
	Web server and version

Agent versions	Data
	Hosted or on-premises Contrast instance

To opt-out of the telemetry feature, set the `CONTRAST_AGENT_TELEMETRY_OPTOUT` environment variable to `1` or `true`.

Telemetry data is securely sent to `telemetry.ruby.contrastsecurity.com`. You can also opt out of telemetry by blocking communication at the network level.

Go agent

The Go agent is a source code rewriter that instruments Go web applications for library support and vulnerability reporting. It provides runtime insight into the source code and libraries that make up the application.

The latest Go agent supports both Assess (IAST) and Protect (RASP) features.



NOTE

As a source code rewriter, installing the Go agent in an application requires access to the application build environment.

As a next step, you can:

- [Install the Go agent \(page 531\)](#)
- [Go supported technologies \(page 530\)](#)

Supported technologies for the Go agent

We support the following technologies for this agent.

Technology	Supported versions	Notes
Language version	<ul style="list-style-type: none">• 1.23• 1.24	<p>Contrast follows the Go release support policy which consists of the two most recently released major versions. Support for Go language versions is shifted as new major versions are released. The application dependencies must be specified in a <code>go.mod</code> file.</p> <p>Not supported:</p> <ul style="list-style-type: none">• 1.22: last supported agent was 7.3.0• 1.21: Last supported agent was 6.14.0• 1.20: Last supported agent was 6.10.0• 1.19: Last supported agent was 5.12.0• 1.18: Last supported agent was 4.2.0• 1.17: Last supported agent was 3.6.0• 1.16: Last supported agent was 3.6.0• 1.15: Last supported agent was 1.12.0
Platforms	<ul style="list-style-type: none">• darwin/amd64• darwin/arm64• linux/amd64• linux/arm64	<p>Contrast builds and tests <code>contrast-go</code> for the listed platforms.</p> <p>You can use <code>contrast-go</code> to cross-compile for other platforms, however, compatibility is not guaranteed.</p>
Dependency management system	Go mod	The application must be part of a Go module . An application can be initialized with modules by running <code>go mod init</code> .

Technology	Supported versions	Notes
Protocol stacks	<ul style="list-style-type: none"> net/http github.com/valyala/fasthttp v1.46.0 and later google.golang.org/grpc v1.17.0 and later 	The agent relies on the protocol implementation to understand application behavior. If the application uses an unsupported protocol package, the agent is not able to report vulnerability or route information.
Routers and frameworks	<ul style="list-style-type: none"> github.com/gofiber/fiber v2.40.0 and later github.com/go-chi/chi/v5 github.com/julienschmidt/httprouter github.com/gin-gonic/gin v1 and later github.com/go-openapi/swag 	<p>The agent can work with an unlisted framework as long as the framework's underlying HTTP implementation is supported. However, some features might be missing or less accurate.</p> <p>For example, the agent might not correctly discover routes registered with an unrecognized router.</p> <p>If your framework is not in the list of supported frameworks or you need assistance, contact Contrast support.</p>
Database support	database/sql	The agent instruments the database/sql package in the standard library to support databases registered as database drivers. See the examples in SQLDrivers .

Install the Go agent

The Go agent uses a tool called `contrast-go` to inject instrumentation into your applications at build time. When you run an instrumented application, the Go agent automatically starts and monitors the application's execution to detect vulnerabilities.



TIP

To see a list of available flags with [command line arguments for contrast-go](#), type `contrast-go -h`.

Steps

1. Install `contrast-go` with the [installer](#):

```
go run github.com/contrast-security-oss/contrast-go-installer@latest latest
```

2. Build your application with `contrast-go`:

```
contrast-go build -o output-name-of-application
```

3. [Configure the Go agent \(page 534\)](#) using the [Go YAML template \(page 535\)](#) or environment variables.
4. Run your application using the executable you generated in step 2.
5. Exercise and test your application.
6. Use the Contrast web interface to explore findings that the agent reports, such as vulnerabilities and library usage information.
By default, your application name is based on the application's Go module. Use search in the Applications list to quickly find your application.

Install the Go agent in a container

Installing the Go agent in a container is essentially the same as the standard installation procedure, except that the installation occurs in a container and, to follow best practices, you should use environment variables to configure the Contrast credentials.

Using environment variables is the most secure method for installing the Go agent in a container. Since containers often migrate through QA and production systems, it's a best practice to avoid hard-coding credentials in the container definition.

**TIP**

If you would like to explore a sample application using the Go agent in a Dockerfile, see the [Go Test Bench project](#).

Before you begin

- You should have a basic understanding of how containers and related software work.
- You may need to adjust the instructions to meet your specific circumstances.

Install, build, and run the Go application

1. Ensure Go is installed.
2. Install `contrast-go` with this command:

```
RUN github.com/contrast-security-oss/contrast-go-installer@latest latest
```

3. Build the application by replacing your normal `go build` command with `contrast-go build`. This step builds an executable with Contrast embedded in it.

```
RUN contrast-go build ./app
```

4. [Configure the agent \(page 534\)](#) with environment variables.

Docker example

This example shows how to install, build, and run a Go application in a Docker container.

```
# Step 1: Install Go. You can use a different base image than the one shown
in
# this example.
FROM golang:1.21 AS builder
WORKDIR /build

COPY . .

# Step 2: This step installs contrast-go and makes sure it's in your $PATH
so
# you can use it in the next step.
RUN go run github.com/contrast-security-oss/contrast-go-installer@latest
latest

# Step 3: This step is your normal build step, but uses contrast-go instead
of
# go. This step doesn't replace Go; it just wraps it so that it can add
# instrumentation during the build process.
RUN contrast-go build ./app

# Optional: Move the finished build to a new container.
# Not required, but nice to have!
FROM alpine:latest
```



```
COPY --from=builder /build/app .  
  
# Step 4: Configure the agent using environment variables.  
  
ENTRYPOINT [ "./app" ]
```

Example of environment variable configuration



NOTE

The Export option in the [Contrast agent configuration editor \(page 109\)](#) is an easy way to create the environment variables for the Contrast credentials.

The process to set environment variables when using a cloud provider typically involves using a secrets manager and then linking the values of those secrets to the environment variables.

For example, you could use this command to build your container:

```
docker build -t my-app-image
```

And then, use these commands when you run the container:

```
docker run -p 3000:3000 --name my-app-instance \  
-e "CONTRAST__API__URL=your-ts-url" \  
-e "CONTRAST__API__API_KEY=your-api-key" \  
-e "CONTRAST__API__SERVICE_KEY=your-service-key" \  
-e "CONTRAST__API__USER_NAME=your-user-name" \  
my-app-image
```

See also

- [Kubernetes and Contrast](#)
- [AWS Fargate and Contrast agents](#)

Install the Go agent with direct download

To install the Go agent:

1. Download the executable files from <https://pkg.contrastsecurity.com>.
The `contrast-go` executables can be downloaded directly for Mac and Linux operating systems. You can see available versions in the [go-agent-release](#) user interface. Replace `<version>` with the version number you want, or `latest`.

- **For Mac:**

```
wget https://pkg.contrastsecurity.com/go-agent-release/<version>/darwin-amd64/contrast-go
```

or

```
curl -L https://pkg.contrastsecurity.com/go-agent-release/<version>/darwin-amd64/contrast-go > contrast-go
```

- **For Linux:**

```
wget https://pkg.contrastsecurity.com/go-agent-release/<version>/linux-  
amd64/contrast-go
```

or

```
curl -L https://pkg.contrastsecurity.com/go-agent-release/<version>/  
linux-amd64/contrast-go > contrast-go
```

2. After download, verify that the agent is executable. For example:

```
chmod u+x contrast-go
```

3. Be sure the application has a `go.mod` file to indicate required dependencies. In the application source directory run the following command:

```
go mod init
```

4. Build your application:

```
./contrast-go build -o output-name-of-application
```

5. [Configure the Go agent \(page 534\)](#) using the [Go YAML template \(page 535\)](#) or environment variables.
6. Run your application using the executable from the output above.
7. Exercise and test your application.
8. Verify that Contrast sees your application.

Configure the Go agent

To configure the agent, the recommended method is to specify [environment variables \(page 110\)](#). Using environment variables is useful if you are using Contrast in containers or CI/CD pipelines.

You can also specify settings in a YAML file called `contrast_security.yaml`.

The simplest way to get started is to use the Go Agent wizard in the Contrast web interface. The agent wizard pre-populates environment variables and the YAML file with the required settings for your organization.



TIP

If you set values with environment variables and in the agent YAML file, the agent uses the environment variables, as described in [order of precedence \(page 106\)](#).

The variables you can set are:

- **Agent Token:** This variable is a base64 encoded JSON object containing the `url`, `api_key`, `service_key`, and `user_name` configuration settings, allowing you to set them set in a single variable.

```
CONTRAST__API__TOKEN
```

If your agent configuration refers to both the legacy settings and the agent token, (in environment variables or the YAML file), the legacy settings take precedence. To use just the agent token value, make sure you remove references to the legacy settings.

- **Legacy settings:** If you are using a Go agent version earlier than 6.11.0, the required variables are:

```
CONTRAST__API__URL  
CONTRAST__API__API_KEY
```

```
CONTRAST__API__SERVICE_KEY
CONTRAST__API__USER_NAME
```

You can also [find the keys \(page 104\)](#) under **Organization settings > Agent** in the Contrast web interface.

Location of the Go configuration file

If you are using the `contrast_security.yaml` file for configuration, the Go agent looks for it in the following directories until it finds one:

- Current directory
- `/etc/contrast/go/`
- `/etc/contrast/`
- **Darwin:**`$HOME/Library/Preferences/contrast/`
- **Darwin:**`$HOME/Library/Preferences/contrast/go/`
- **Linux:**`$XDG_CONFIG_DIR/contrast/`
- **Linux:**`$XDG_CONFIG_DIR/contrast/go/`

Go YAML template

This template includes all available settings for the Go agent. (Learn more about [YAML configuration \(page 107\)](#).)

You can also find the settings and generate a custom configuration file with the [Contrast agent configuration editor \(page 109\)](#).

```
#
=====
==
# Use the properties in this YAML file to configure a Contrast agent.
# Go to https://docs.contrastsecurity.com/en/order-of-precedence.html
# to determine the order of precedence for configuration values.
#
=====
==

# Use this setting if you want to temporarily disable a Contrast agent.
# Set to `true` to enable the agent; set to `false` to disable the agent.
# enable: true

#
=====
==
# api
# Use the properties in this section to connect the agent to the Contrast
# UI.
#
=====
==
api:

# ***** REQUIRED *****
# Set the URL for the Contrast UI.
```

```
url: https://app.contrastsecurity.com/Contrast

# ***** REQUIRED *****
# Set the API key needed to communicate with the Contrast UI.
api_key: NEEDS_TO_BE_SET

# ***** REQUIRED *****
# Set the service key needed to communicate with the Contrast
# UI. It is used to calculate the Authorization header.
service_key: NEEDS_TO_BE_SET

# ***** REQUIRED *****
# Set the user name used to communicate with the Contrast
# UI. It is used to calculate the Authorization header.
user_name: NEEDS_TO_BE_SET

# base64 encoded JSON object containing the `url`,
# `api_key`, `service_key`, and `user_name` config options,
# allowing them all to be set in a single variable.
# token: NEEDS_TO_BE_SET

#
=====
# api.certificate
# Use the following properties for communication
# with the Contrast UI using certificates.
#
=====
# certificate:

# If set to `false`, the agent will ignore the
# certificate configuration in this section.
# enable: true

# Set the absolute or relative path to a CA for communication
# with the Contrast UI using a self-signed certificate.
# ca_file: NEEDS_TO_BE_SET

#
=====
# api.proxy
# Use the following properties for communication
# with the Contrast UI over a proxy.
#
=====
# proxy:

# Set value to `true` for the agent to communicate
# with the Contrast web interface over a proxy. Set
# value to `false` if you don't want to use the proxy.
# enable: NEEDS_TO_BE_SET

# Set the proxy host. It must be set with port and scheme.
# host: localhost
```

```
# Set the proxy port. It must be set with host and scheme.
# port: 1234

# Set the proxy scheme (e.g., `http` or
# `https`). It must be set with host and port.
# scheme: http

# Set the URL for your Proxy Server. The URL form is `scheme://
host:port`.
# url: NEEDS_TO_BE_SET

# Set the proxy user.
# user: NEEDS_TO_BE_SET

# Set the proxy password.
# pass: NEEDS_TO_BE_SET

#
=====
==
# agent
# Use the properties in this section to control the way and frequency
# with which the agent communicates to logs and the Contrast UI.
#
=====
==
# agent:

#
=====
# agent.logger
# Define the following properties to set logging values.
# If the following properties are not defined, the
# agent uses the logging values from the Contrast UI.
#
=====
# logger:

# Enable diagnostic logging by setting a path to a log file.
# While diagnostic logging hurts performance, it generates
# useful information for debugging Contrast. The value set here
# is the location to which the agent saves log output. If no
# log file exists at this location, the agent creates a file.
#
# Example - `/opt/Contrast/contrast.log` creates a log in the
# `/opt/Contrast` directory, and rotates it automatically as needed.
#
# path: ./contrast_agent.log

# Set the the log output level. Valid options are
# `ERROR`, `WARN`, `INFO`, `DEBUG`, and `TRACE`.
# level: INFO

# Set to `true` to redirect all logs to
# `stdout` instead of the file system.
```

```
# stdout: false

#
=====
# agent.security_logger
# Define the following properties to set security logging
# values associated with Protect. If not defined, the agent
# uses the security logging (CEF) values from the Contrast UI.
#
=====
# security_logger:

# Set the file to which the agent logs security events.
# path: ./contrast/security.log

# Set the log level for security logging. Valid options
# are `ERROR`, `WARN`, `INFO`, `DEBUG`, and `TRACE`.
# level: ERROR

#
=====
# agent.go
# The following properties apply to any Go agent-wide configurations.
#
=====
# go:

#
=====
# agent.go.preview
# Enable opt-in Go agent features.
#
=====
# preview:

# Enable Assess gRPC sources.
# grpc: false

#
=====
# agent.go.profile
# Enable Go agent self-profiling features.
#
=====
# profile:

# Enable CPU profiling for running application.
# cpu: false

# Enable memory profiling for running application.
# mem: false

#
=====
==
```

```
# inventory
# Use the properties in this section to override the inventory features.
#
=====
==
# inventory:

# Set to `false` to disable inventory features in the agent.
# enable: true

# Set to `false` to disable library analysis.
# analyze_libraries: true

#
=====
==
# assess
# Use the properties in this section to control Assess.
#
=====
==
# assess:

# Include this property to determine if the Assess
# feature should be enabled. If this property is not
# present, the decision is delegated to the Contrast UI.
# enable: false

# Apply a list of labels to vulnerabilities and preflight
# messages. Labels must be formatted as a comma-delimited list.
# Example - `label1, label2, label3`
#
# tags: NEEDS_TO_BE_SET

#
=====
# assess.rules
# Use the following properties to control simple rule configurations.
#
=====
# rules:

# Define a list of Assess rules to disable in the agent. To view a
# list of rule names, in Contrast go to user menu > Policy Management >
# Assess rules. The rules must be formatted as a comma-delimited list.
#
# Example - Set `reflected-xss,sql-injection` to disable
# the reflected-xss rule and the sql-injection rule.
#
# disabled_rules: NEEDS_TO_BE_SET

#
=====
==
# protect
```

```
# Use the properties in this section to override Protect features.
#
=====
==
# protect:

# Include this property to determine if the Protect
# feature should be enabled. If this property is not
# present, the decision is delegated to the Contrast UI.
# enable: false

#
=====
# protect.probe_analysis
# Use the settings in this section to
# control the behavior of probe analysis.
#
=====
# probe_analysis:

# Set to `false` to disable probe analysis.
# enable: true

#
=====
==
# application
# Use the properties in this section for
# the application(s) hosting this agent.
#
=====
==
# application:

# Override the reported application name.
#
# Note - On systems where multiple, distinct applications may be served
# by a single process, this configuration causes the agent to report
# all discovered applications as one application with the given name.
#
# Note - On Java systems where multiple, distinct applications may be
# served by a single process, this configuration causes the agent to
report
# all discovered applications as one application with the given name.
#
# name: NEEDS_TO_BE_SET

# Override the reported application path.
# path: NEEDS_TO_BE_SET

# Add the name of the application group with which this
# application should be associated in the Contrast UI.
# group: NEEDS_TO_BE_SET

# Add the application code this application should use in the Contrast UI.
```



```
# code: NEEDS_TO_BE_SET

# Override the reported application version.
# version: NEEDS_TO_BE_SET

# Apply labels to an application. Labels must
# be formatted as a comma-delimited list.
# Example - `label1,label2,label3`
#
# tags: NEEDS_TO_BE_SET

# Define a set of `key=value` pairs (which conforms to RFC 2253) for
# specifying user-defined metadata associated with the application. The
# set must be formatted as a comma-delimited list of `key=value` pairs.
# Example - `business-unit=accounting, office=Baltimore`
#
# metadata: NEEDS_TO_BE_SET

# Provide the ID of a session which already exists in the Contrast
# UI. Vulnerabilities discovered by the agent are associated with
# this session. If an invalid ID is supplied, the agent will be
# disabled. This option and `application.session_metadata` are
# mutually exclusive; if both are set, the agent will be disabled.
# session_id: NEEDS_TO_BE_SET

# Provide metadata which is used to create a new session ID in the
# Contrast UI. Vulnerabilities discovered by the agent are associated with
# this new session. This value should be formatted as `key=value` pairs
# (conforming to RFC 2253). Available key names for this configuration
# are branchName, buildNumber, commitHash, committer, gitTag, repository,
# testRun, and version. This option and `application.session_id` are
# mutually exclusive; if both are set the agent will be disabled.
# session_metadata: NEEDS_TO_BE_SET

#
=====
==
# server
# Use the settings in this section to set metadata for the server
# hosting this agent. Contrast recognizes common, supported server
# names, paths, types and environments. Doing this may require a new
# server or license, and it may affect functionality of some features.
#
=====
==
# server:

# Override the reported server name.
# name: localhost

# Override the reported server path.
# path: NEEDS_TO_BE_SET

# Override the reported server type.
# type: NEEDS_TO_BE_SET
```

```
# Set the environment directly to override the default set
# by the Contrast UI. This allows the user to configure the
# environment dynamically at startup rather than manually
# updating the Server in the Contrast UI themselves afterwards.
#
# Valid values include `QA`, `PRODUCTION` and `DEVELOPMENT`.
# For example, `PRODUCTION` registers this Server as
# running in a `PRODUCTION` environment, regardless of the
# organization's default environment in the Contrast UI.
#
# environment: NEEDS_TO_BE_SET

# Apply a list of labels to the server. Labels
# must be formatted as a comma-delimited list.
# Example - `label1,label2,label3`
#
# tags: NEEDS_TO_BE_SET
```

```
#
=====
==
# Use the properties in this YAML file to configure a Contrast agent.
# Go to https://docs.contrastsecurity.com/en/order-of-precedence.html
# to determine the order of precedence for configuration values.
#
=====
==

# Use this setting if you want to temporarily disable a Contrast agent.
# Set to `true` to enable the agent; set to `false` to disable the agent.
# enable: true

#
=====
==
# api
# Use the properties in this section to connect the agent to the Contrast
# UI.
#
=====
==
api:

# ***** REQUIRED *****
# Set the URL for the Contrast UI.
url: https://app.contrastsecurity.com/Contrast

# ***** REQUIRED *****
# Set the API key needed to communicate with the Contrast UI.
api_key: NEEDS_TO_BE_SET

# ***** REQUIRED *****
# Set the service key needed to communicate with the Contrast
```

```
# UI. It is used to calculate the Authorization header.
service_key: NEEDS_TO_BE_SET

# ***** REQUIRED *****
# Set the user name used to communicate with the Contrast
# UI. It is used to calculate the Authorization header.
user_name: NEEDS_TO_BE_SET

# base64 encoded JSON object containing the `url`,
# `api_key`, `service_key`, and `user_name` config options,
# allowing them all to be set in a single variable.
# token: NEEDS_TO_BE_SET

#
=====
# api.certificate
# Use the following properties for communication
# with the Contrast UI using certificates.
#
=====
# certificate:

# If set to `false`, the agent will ignore the
# certificate configuration in this section.
# enable: true

# Set the absolute or relative path to a CA for communication
# with the Contrast UI using a self-signed certificate.
# ca_file: NEEDS_TO_BE_SET

#
=====
# api.proxy
# Use the following properties for communication
# with the Contrast UI over a proxy.
#
=====
# proxy:

# Set value to `true` for the agent to communicate
# with the Contrast web interface over a proxy. Set
# value to `false` if you don't want to use the proxy.
# enable: NEEDS_TO_BE_SET

# Set the proxy host. It must be set with port and scheme.
# host: localhost

# Set the proxy port. It must be set with host and scheme.
# port: 1234

# Set the proxy scheme (e.g., `http` or
# `https`). It must be set with host and port.
# scheme: http

# Set the URL for your Proxy Server. The URL form is `scheme://`
```

```
host:port`.
# url: NEEDS_TO_BE_SET

# Set the proxy user.
# user: NEEDS_TO_BE_SET

# Set the proxy password.
# pass: NEEDS_TO_BE_SET

#
=====
==
# agent
# Use the properties in this section to control the way and frequency
# with which the agent communicates to logs and the Contrast UI.
#
=====
==
# agent:

#
=====
# agent.logger
# Define the following properties to set logging values.
# If the following properties are not defined, the
# agent uses the logging values from the Contrast UI.
#
=====
# logger:

# Enable diagnostic logging by setting a path to a log file.
# While diagnostic logging hurts performance, it generates
# useful information for debugging Contrast. The value set here
# is the location to which the agent saves log output. If no
# log file exists at this location, the agent creates a file.
#
# Example - `/opt/Contrast/contrast.log` creates a log in the
# `/opt/Contrast` directory, and rotates it automatically as needed.
#
# path: ./contrast_agent.log

# Set the the log output level. Valid options are
# `ERROR`, `WARN`, `INFO`, `DEBUG`, and `TRACE`.
# level: INFO

# Set to `true` to redirect all logs to
# `stdout` instead of the file system.
# stdout: false

#
=====
# agent.security_logger
# Define the following properties to set security logging
# values associated with Protect. If not defined, the agent
# uses the security logging (CEF) values from the Contrast UI.
```

```
#
=====
# security_logger:

# Set the file to which the agent logs security events.
# path: ./contrast/security.log

# Set the log level for security logging. Valid options
# are `ERROR`, `WARN`, `INFO`, `DEBUG`, and `TRACE`.
# level: ERROR

#
=====
# agent.go
# The following properties apply to any Go agent-wide configurations.
#
=====
# go:

#
=====
# agent.go.preview
# Enable opt-in Go agent features.
#
=====
# preview:

# Enable Assess gRPC sources.
# grpc: false

#
=====
# agent.go.profile
# Enable Go agent self-profiling features.
#
=====
# profile:

# Enable CPU profiling for running application.
# cpu: false

# Enable memory profiling for running application.
# mem: false

#
=====
==
# inventory
# Use the properties in this section to override the inventory features.
#
=====
==
# inventory:

# Set to `false` to disable inventory features in the agent.
```

```
# enable: true

# Set to `false` to disable library analysis.
# analyze_libraries: true

#
=====
==
# assess
# Use the properties in this section to control Assess.
#
=====
==
# assess:

# Include this property to determine if the Assess
# feature should be enabled. If this property is not
# present, the decision is delegated to the Contrast UI.
# enable: false

# Apply a list of labels to vulnerabilities and preflight
# messages. Labels must be formatted as a comma-delimited list.
# Example - `label1, label2, label3`
#
# tags: NEEDS_TO_BE_SET

#
=====
# assess.rules
# Use the following properties to control simple rule configurations.
#
=====
# rules:

# Define a list of Assess rules to disable in the agent. To view a
# list of rule names, in Contrast go to user menu > Policy Management >
# Assess rules. The rules must be formatted as a comma-delimited list.
#
# Example - Set `reflected-xss,sql-injection` to disable
# the reflected-xss rule and the sql-injection rule.
#
# disabled_rules: NEEDS_TO_BE_SET

#
=====
==
# protect
# Use the properties in this section to override Protect features.
#
=====
==
# protect:

# Include this property to determine if the Protect
# feature should be enabled. If this property is not
```

```
# present, the decision is delegated to the Contrast UI.
# enable: false

#
=====
# protect.probe_analysis
# Use the settings in this section to
# control the behavior of probe analysis.
#
=====
# probe_analysis:

# Set to `false` to disable probe analysis.
# enable: true

#
=====
==
# application
# Use the properties in this section for
# the application(s) hosting this agent.
#
=====
==
# application:

# Override the reported application name.
#
# Note - On systems where multiple, distinct applications may be served
# by a single process, this configuration causes the agent to report
# all discovered applications as one application with the given name.
#
# Note - On Java systems where multiple, distinct applications may be
# served by a single process, this configuration causes the agent to
report
# all discovered applications as one application with the given name.
#
# name: NEEDS_TO_BE_SET

# Override the reported application path.
# path: NEEDS_TO_BE_SET

# Add the name of the application group with which this
# application should be associated in the Contrast UI.
# group: NEEDS_TO_BE_SET

# Add the application code this application should use in the Contrast UI.
# code: NEEDS_TO_BE_SET

# Override the reported application version.
# version: NEEDS_TO_BE_SET

# Apply labels to an application. Labels must
# be formatted as a comma-delimited list.
# Example - `label1,label2,label3`
```

```
#
# tags: NEEDS_TO_BE_SET

# Define a set of `key=value` pairs (which conforms to RFC 2253) for
# specifying user-defined metadata associated with the application. The
# set must be formatted as a comma-delimited list of `key=value` pairs.
# Example - `business-unit=accounting, office=Baltimore`
#
# metadata: NEEDS_TO_BE_SET

# Provide the ID of a session which already exists in the Contrast
# UI. Vulnerabilities discovered by the agent are associated with
# this session. If an invalid ID is supplied, the agent will be
# disabled. This option and `application.session_metadata` are
# mutually exclusive; if both are set, the agent will be disabled.
# session_id: NEEDS_TO_BE_SET

# Provide metadata which is used to create a new session ID in the
# Contrast UI. Vulnerabilities discovered by the agent are associated with
# this new session. This value should be formatted as `key=value` pairs
# (conforming to RFC 2253). Available key names for this configuration
# are branchName, buildNumber, commitHash, committer, gitTag, repository,
# testRun, and version. This option and `application.session_id` are
# mutually exclusive; if both are set the agent will be disabled.
# session_metadata: NEEDS_TO_BE_SET

#
=====
==
# server
# Use the settings in this section to set metadata for the server
# hosting this agent. Contrast recognizes common, supported server
# names, paths, types and environments. Doing this may require a new
# server or license, and it may affect functionality of some features.
#
=====
==
# server:

# Override the reported server name.
# name: localhost

# Override the reported server path.
# path: NEEDS_TO_BE_SET

# Override the reported server type.
# type: NEEDS_TO_BE_SET

# Set the environment directly to override the default set
# by the Contrast UI. This allows the user to configure the
# environment dynamically at startup rather than manually
# updating the Server in the Contrast UI themselves afterwards.
#
# Valid values include `QA`, `PRODUCTION` and `DEVELOPMENT`.
# For example, `PRODUCTION` registers this Server as
```



```
# running in a `PRODUCTION` environment, regardless of the
# organization's default environment in the Contrast UI.
#
# environment: NEEDS_TO_BE_SET

# Apply a list of labels to the server. Labels
# must be formatted as a comma-delimited list.
# Example - `label1,label2,label3`
#
# tags: NEEDS_TO_BE_SET
```

Contrast service



IMPORTANT

The Contrast service is required for earlier versions of Node.js (before Node.js agent version 5.0.0) and Python (before Python agent version 5.19.0). Newer versions of the agent use a more performant and native analysis that does not require the Contrast Service. Refer to the documentation for each agent to verify the versions where these changes occur.

The Contrast service is a stand-alone executable that enables the communication between Contrast and multi-process dynamic language agents (Node.js and Python agents). It passes settings from Contrast to the agent. It also aggregates and sends information from the agent back to Contrast.

It is compiled for various supported architectures:

- Linux 64-bit
- Macintosh 64-bit
- Windows 64-bit

The service is packaged with the Node.js and Python agents and starts automatically when the instrumented application is started. The service is not packaged or started by the Go agent. You must have a service installed, configured and running for the Go agent to function. You may do the same for more control when running the Node.js or Python agents.

Install the Contrast service

Installation varies depending on your system:

- **Linux:** Install the Contrast service with a system package manager.
- **Debian:** Use the commands to install from the correct Debian repository.

1. Get the `CODENAME` for your Ubuntu release.

```
grep VERSION_CODENAME /etc/os-release
```

2. Update the command below with the `CODENAME`, and run the commands.

```
curl https://pkg.contrastsecurity.com/api/gpg/key/public | sudo apt-  
key add -  
echo "deb https://pkg.contrastsecurity.com/debian-public/ CODENAME  
contrast" | sudo tee /etc/apt/sources.list.d/contrastc.list
```

3. Install the Contrast service:

```
sudo apt-get update && sudo apt-get install contrast-service
```

4. [Configure the Contrast service \(page 550\)](#).

- **Red Hat Package Manager (RPM):** Use these commands to install from Contrast's yum repository.

1. Configure your system to use the repository:

```
OSREL=$(rpmquery -E "%{rhel}")
sudo -E tee /etc/yum.repos.d/contrast.repo << EOF
[contrast]
name=contrast repo
baseurl=https://pkg.contrastsecurity.com/rpm-public/centos-$OSREL/
gpgcheck=0
enabled=1
EOF
```

2. Install the Contrast service:

```
yum install contrast-service
```

3. [Configure the Contrast service. \(page 550\)](#)

**TIP**

To remove the `contrast-service` package, run `apt-get remove contrast-service` or `yum remove contrast-service`.

Configure the Contrast service

The Contrast service is not preconfigured with connection parameters. You must configure the service with a YAML configuration file.

When installed as a system service, the Contrast service is controlled by this YAML configuration file located in the `/etc` directory. Frequently, the service shares the same `contrast_security.yaml` file with any other applications on the same server, to ensure that all connection values (like the socket name or port number) are consistent.

Assuming an application-specific configuration file is not already installed in the application's working directory, the location of the YAML configuration file determines whether it can be shared with the agent on the same server:

- If you **don't** want it to be shared, place the configuration file at `/etc/contrast/webserver/contrast_security.yaml`.
- If you **do** want it to be shared, place the configuration file at `/etc/contrast/contrast_security.yaml`.

A default configuration YAML file is installed with the Contrast service Linux package at `/etc/contrast/webserver/contrast_security.yaml`. This template has placeholders for most necessary items, but you should update the following:

- **api:** Set the API properties. This determines how the Contrast service connects to Contrast.
- **agent:** This is the top-level configuration section for agent-related configuration.
 - **service:** These options affect communication between an agent and the Contrast service. The connection configuration must be identical between the Contrast service and the agent communicating with that service.

- **socket:** The path to the local unix socket (for example, `/tmp/contrast.sock`)
- **host and port:** Optionally, instead of socket, the Contrast Service can be configured to connect at a host and port.
- **grpc:** (applies to Go and Node.js agents only) Set to "true" to use gRPC for agent to service communication. This is optional and may provide a slight performance improvement.

If this configuration has an issue or incorrect values, or the Contrast service fails to connect to Contrast, you can troubleshoot the failed connection result at `/var/log/contrast/service.log`.

Install the Contrast service

Installation varies depending on your system:

- **Linux:** Install the Contrast service with a system package manager.
- **Debian:** Use the commands to install from the correct Debian repository.

1. Get the `CODENAME` for your Ubuntu release.

```
grep VERSION_CODENAME /etc/os-release
```

2. Update the command below with the `CODENAME`, and run the commands.

```
curl https://pkg.contrastsecurity.com/api/gpg/key/public | sudo apt-key add -  
echo "deb https://pkg.contrastsecurity.com/debian-public/ CODENAME  
contrast" | sudo tee /etc/apt/sources.list.d/contrastc.list
```

3. Install the Contrast service:

```
sudo apt-get update && sudo apt-get install contrast-service
```

4. [Configure the Contrast service \(page 550\)](#).

- **Red Hat Package Manager (RPM):** Use these commands to install from Contrast's yum repository.

1. Configure your system to use the repository:

```
OSREL=$(rpmquery -E "%{rhel}")  
sudo -E tee /etc/yum.repos.d/contrast.repo << EOF  
[contrast]  
name=contrast repo  
baseurl=https://pkg.contrastsecurity.com/rpm-public/centos-$OSREL/  
gpgcheck=0  
enabled=1  
EOF
```

2. Install the Contrast service:

```
yum install contrast-service
```

3. [Configure the Contrast service. \(page 550\)](#)



TIP

To remove the `contrast-service` package, run `apt-get remove contrast-service` or `yum remove contrast-service`.

Configure the Contrast service

The Contrast service is not preconfigured with connection parameters. You must configure the service with a YAML configuration file.

When installed as a system service, the Contrast service is controlled by this YAML configuration file located in the `/etc` directory. Frequently, the service shares the same `contrast_security.yaml` file with any other applications on the same server, to ensure that all connection values (like the socket name or port number) are consistent.

Assuming an application-specific configuration file is not already installed in the application's working directory, the location of the YAML configuration file determines whether it can be shared with the agent on the same server:

- If you **don't** want it to be shared, place the configuration file at `/etc/contrast/webserver/contrast_security.yaml`.
- If you **do** want it to be shared, place the configuration file at `/etc/contrast/contrast_security.yaml`.

A default configuration YAML file is installed with the Contrast service Linux package at `/etc/contrast/webserver/contrast_security.yaml`. This template has placeholders for most necessary items, but you should update the following:

- **api:** Set the API properties. This determines how the Contrast service connects to Contrast.
- **agent:** This is the top-level configuration section for agent-related configuration.
 - **service:** These options affect communication between an agent and the Contrast service. The connection configuration must be identical between the Contrast service and the agent communicating with that service.
 - **socket:** The path to the local unix socket (for example, `/tmp/contrast.sock`)
 - **host and port:** Optionally, instead of socket, the Contrast Service can be configured to connect at a host and port.
 - **grpc:** (applies to Go and Node.js agents only) Set to "true" to use gRPC for agent to service communication. This is optional and may provide a slight performance improvement.

If this configuration has an issue or incorrect values, or the Contrast service fails to connect to Contrast, you can troubleshoot the failed connection result at `/var/log/contrast/service.log`.

Agent Operator (Kubernetes operator)

The Contrast Agent Operator is a standard [Kubernetes operator](#) that executes within Kubernetes and OpenShift clusters to automate injecting Contrast agents into existing workloads, configuring injected agents, and facilitating agent upgrades.



NOTE

The Contrast Agent Operator is an open-source project. You can review the code and contribute to its development [here](#).

Security policies

The Contrast Agent Operator supports clusters with various security policies. It is recommended to familiarize yourself with them before installation. See [here](#) for the latest policies.

Custom registries

If you want to use Agent Operator in an air-gapped environment (or you cannot use DockerHub), use the [📄 Contrast custom registry examples](#) to guide you.

Pod restart

Configuration changes are the primary drivers for [pod restarts \(page 572\)](#) and affect the pod's runtime environment.

Next steps

To get started, [install and configure \(page 554\)](#) the agent operator.

Familiarize yourself with the [Agent Operator supported technologies \(page 553\)](#) and [Agent Operator networking requirements \(page 554\)](#).

See also

- [Agent Operator telemetry \(page 572\)](#)

Supported technologies for Agent Operator

Kubernetes / OpenShift support

For the latest supported technology information see [here](#).

Kubernetes version	OpenShift version	Operator version	End-of-support
v1.31		v0.14.0+	2025-10-28
v1.30		v0.14.0+	2025-06-28
v1.29	v4.16	v0.14.0+	2025-02-28
v1.28	v4.15	v0.14.0+	2024-10-28
v1.27	v4.14	v0.14.0+	2024-06-28

- The Contrast Agent Operator follows the upstream [Kubernetes community support policy](#). End-of-life dates are documented on the [Kubernetes releases](#) page.
- OpenShift support is dependent on the included version of Kubernetes. For example, OpenShift v4.10 uses Kubernetes v1.23 and will be supported by Contrast until 2023-02-28. See Red Hat's [support article](#) for the mapping between Kubernetes and OpenShift versions.
- The Contrast Agent Operator only supports executing on Linux amd64/arm64 hosts and will refuse to be scheduled onto incompatible nodes. Additionally, the operator only supports injecting workloads running on Linux amd64/arm64 hosts, even if the Contrast Agent supports additional platforms. Contact [Contrast Support](#) if Kubernetes on Windows is desired.

Agent types

Agent	Agent type	Support status	Compatibility notes
.NET Core	dotnet-core	Supported	Supported .NET Core technologies (page 271)
Java	java	Supported	Supported Java technologies (page 120)
NodeJS	nodejs	Supported	Supported Node.js technologies (page 332)
	OR		
	nodejs-esm		
PHP	php	Beta	Supported PHP technologies (page 391)
Python	python	Supported	Supported Python technologies (page 410)

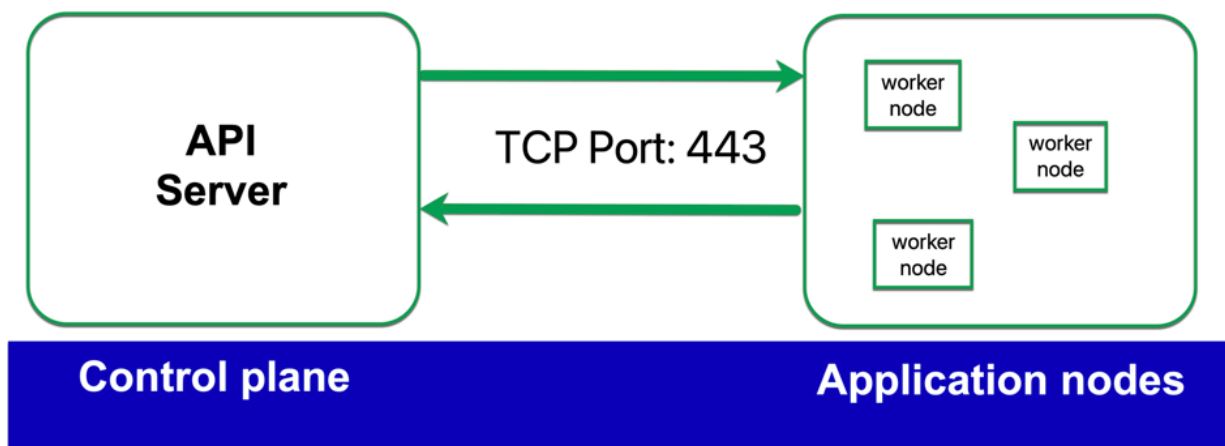
**NOTE**

- Injection of the Node.js agent may result in a substantial increase in the startup time of the instrumented application. If startup time is unacceptable, injecting the agent during compilation may be desirable. If the application was injected by the Node.js agent during compilation then injection during runtime by the operator should be disabled. See the [rewriter CLI \(page 387\)](#) for more information.
- NodeJS ESM injection is only supported on NodeJS LTS versions $\geq 18.19.0$.
- Injection of a PHP application is in beta. Beta status means the feature might change or act unexpectedly. Using this feature, you agree to the [Contrast Beta Terms and Conditions \(page 1275\)](#).

Agent Operator networking requirements

The agent operator needs to be installed in environments where the networking restrictions do not block functionality.

To maintain optimal functionality, it is required to set up bi-directional communication between the control plane and worker nodes on port 443.



Install the Agent Operator

Getting the agent operator synced with Contrast can be divided into a few areas, so this guide should get the agent operator up and running in just a few minutes so you can see how it works.

Set up the agent operator using one of the methods described below.

- With [Helm charts \(page 555\)](#). This is the recommended method.
- With [Terraform \(page 557\)](#)
- With a [Manifest file \(page 559\)](#)

The operator is configured using declarative Kubernetes native resource types. Resource types are documented in the [agent operator configuration \(page 560\)](#) section.

If you prefer, use the [agent operator walkthrough \(page 566\)](#) as an alternative method for installation and configuration.

See also

- [Agent Operator networking requirements \(page 554\)](#)
- [Agent Operator telemetry \(page 572\)](#)

Set up Agent Operator with Helm charts

Helm is a package manager for Kubernetes, which helps manage Kubernetes applications. Helm uses charts to configure, install, and upgrade Kubernetes Operators. This is the recommended method of installation.

Before you begin

Make sure you have everything you need before you start:

- Must use [supported versions, frameworks, and tools \(page 553\)](#).

Steps

1. Run these Helm commands:

```
helm repo add contrast https://contrastsecurity.dev/helm-charts
helm repo update contrast
helm show values contrast/contrast-agent-operator > contrast-agent-operator.yaml
```

2. [Add agent keys \(page 104\)](#) to the `clusterDefaults` section of the YAML file. You will also need to set the `enabled: property` to `true`.

```
clusterDefaults:
  enabled: true
  url: YOUR_CONTRAST_URL
  apiKeyValue: YOUR_API_KEY
  serviceKeyValue: YOUR_AGENT_SERVICE_KEY
  userNameValue: YOUR_AGENT_USERNAME
  yaml: |-
    enable: true
```

3. Run this Helm command:

```
helm upgrade --install --namespace contrast-agent-operator --create-namespace -f contrast-agent-operator.yaml contrast-agent-operator contrast/contrast-agent-operator
```

The Helm notes include details on the labels to apply to your workloads, as shown in this example:

```
Release "contrast-agent-operator" has been upgraded. Happy Helming!
NAME: contrast-agent-operator
LAST DEPLOYED: Tue Jul  2 12:04:40 2024
NAMESPACE: default
STATUS: deployed
REVISION: 4
TEST SUITE: None
NOTES:
contrast-agent-operator version 1.4.0 deployed!
6 injectors have been deployed to namespace: default
To use with your workloads:

contrast-java-injector (java):
  kubectl label deployment/<your_deployment_name> contrast-agent=java
```

```

contrast-dotnet-core-injector (dotnet-core):
    kubectl label deployment/<your_deployment_name> contrast-
agent=dotnet-core

contrast-nodejs-injector (nodejs):
    kubectl label deployment/<your_deployment_name> contrast-
agent=nodejs

contrast-nodejs-esm-injector (nodejs-esm):
    kubectl label deployment/<your_deployment_name> contrast-
agent=nodejs-esm

contrast-php-injector (php):
    kubectl label deployment/<your_deployment_name> contrast-agent=php

contrast-python-injector (python):
    kubectl label deployment/<your_deployment_name> contrast-
agent=python

Cluster agent defaults deployed

To watch the operator logs:
    kubectl logs -f -l app.kubernetes.io/part-of=contrast-agent-
operator --namespace contrast-agent-operator

More documentation: https://docs.contrastsecurity.com/en/agent-
operator.html

Get support: https://support.contrastsecurity.com /
support@contrastsecurity.com

```

- Label your deployments by using the [values from the table \(page 558\)](#).

**TIP**

Run the `kubectl get deployments` command to also find the deployment names.

Example commands for labeling a deployment:

Run:

```
kubectl get deployments
```

Get an output (for example):

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
app1-deployment	0 / 3	0	0	1s

Then run the following command:

```
kubectl label deployment app1-deployment contrast-agent=java
```

The default configuration from the YAML file deploys AgentInjectors only to the default namespace. If you use other namespaces, you can add them to the `agentInjectors.namespaces` array in the YAML file.

The following example shows how you might configure the `agentInjectors.namespaces` array:


```
agentInjectors:
  enabled: true
  # Required. All injectors will be created in each specified namespace.
  lookupNamespaces:
    # If enabled, Helm will lookup namespaces and deploy AgentInjectors
    to any accessible namespaces.
    deployToAllAccessibleNamespaces: true
    # List of namespace patterns to exclude deploying AgentInjectors to
    only when looking up namespaces.
    excludePatterns:
      - gatekeeper*
      - kube*
    # Required if lookupNamespaces.deployToAllAccessibleNamespaces is not
    enabled. All injectors will be created in each specified namespace.
    namespaces:
      - default
    injectors:
      ...
```

Set up Agent Operator with Terraform

Terraform can be used to create code that provisions all sorts of resources including Kubernetes clusters.

Before you begin

Make sure you have everything you need before you start:

- Must use [supported versions, frameworks, and tools \(page 553\)](#).

Steps

1. Create a values file based on the [latest version of the YAML file found here](#).
2. Name it `contrast-agent-operator.yaml`.
3. [Add agent keys \(page 104\)](#) to the `clusterDefaults` section of the YAML file. You will also need to set the `enabled:` property to `true`.

```
clusterDefaults:
  enabled: true
  url: YOUR_CONTRAST_URL
  apiKeyValue: YOUR_API_KEY
  serviceKeyValue: YOUR_AGENT_SERVICE_KEY
  userNameValue: YOUR_AGENT_USERNAME
  yaml: |-
    enable: true
```

4. Label your deployments by using the [values from the table \(page 558\)](#).



TIP

Run the `kubectl get deployments` command to also find the deployment names.

Example commands for labeling a deployment via CLI:

Run:

```
kubectl get deployments
```

Get an output (for example):

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
appl-deployment	0/3	0	0	1s

Then run the following command:

```
kubectl label deployment appl-deployment contrast-agent=java
```

Updating your Helm chart for the application being deployed is highly recommended.

5. Add the following Terraform code:

```
resource "helm_release" "contrast-agent-operator" {  
  name      = "contrast-agent-operator"  
  repository = "https://contrastsecurity.dev/helm-charts"  
  chart     = "contrast-agent-operator"  
  values = [  
    file("${path.module}/contrast-agent-operator.yaml")  
  ]  
}
```

You can use the [values in this table \(page 558\)](#).



NOTE

If the YAML file is edited any time after this setup, you will need to repeat all of the steps above to get the Operator properly running again. Run the `terraform apply` command after step 5.

Values and labels for the Operator

Label selectors

The following is the default mapping for label selectors used by the Operator.

Name	Value	Instrumentation
contrast-agent	java	Adds the Java agent to the container
contrast-agent	dotnet-core	Adds the .NET Core agent to the container
contrast-agent	nodejs	Adds the Node.js agent to the container
	OR	
	nodejs-esm	
contrast-agent	php	Adds the PHP agent to the container
contrast-agent	python	Adds the Python agent to the container

Terraform values

Terraform values include the following.

Key	Definition
namespace	Namespace where Operator is installed.
image.registry	Registry to pull the image from. Defaults to Contrast's registry.
image.repository	The repository to pull the image from. Defaults to Contrast's repository.
image.tag	Tag to pull from. Defaults to the version of the chart.
operator.defaultRegistry	Same as <code>image.registry</code> .

Key	Definition
<code>operator.eventQueueSize</code>	The operator processes events from a stream in Kubernetes. This is the maximum number of messages to process.
<code>operator.eventQueueFullMode</code>	Wait or DropOldest.
<code>operator.webhookSecretName</code>	The name of the secret. Unless the secret was manually created, leave it as default.
<code>operator.webhookConfiguration</code>	The name of the webhook configuration. Unless the configuration was manually created, leave it as default.
<code>operator.enableEarlyChaining</code>	Whether or not early chaining should be added (when using the agent with other agents).
<code>clusterDefaults.enabled</code>	Whether or not the cluster defaults are enabled. This is recommended to be enabled to consolidate API keys.
<code>defaultsDefaults.url</code>	The agent hostname to use from the Kubernetes cluster.
<code>clusterDefaults.apiKeyValue</code> <code>serviceKeyValue</code> <code>userNameValue</code>	The values for the agent user to use. Should be referenced as secrets on the cluster.
<code>clusterDefaults.yaml</code>	Optional additional YAML as generated by https://agent.config.contrastsecurity.com/ . Omit the API section.

Install Agent Operator with manifest files

Contrast provides a single-file installation YAML that can be directly applied to a cluster and provides reasonable defaults. Additional modifications may be desired based on your specific circumstances.

Before you begin

Make sure you have everything you need before you start:

- Must use [supported versions, frameworks, and tools \(page 553\)](#).

Steps



NOTE

It is possible to install into a namespace other than the default `contrast-agent-operator`, although modifications to the deployment manifests will be required.

1. Executing as a cluster administrator, apply the operator manifests using `kubectl` (Kubernetes) or `oc` (OpenShift).

```
kubectl apply -f https://github.com/Contrast-Security-OSS/agent-operator/releases/latest/download/install-prod.yaml
```

```
oc apply -f https://github.com/Contrast-Security-OSS/agent-operator/releases/latest/download/install-prod.yaml
```

The manifests:

- Create the `contrast-agent-operator` namespace.
 - Install the operator Deployment workload.
 - Install the required Custom Resource Definitions.
 - Configure RBAC with the minimum necessary permissions.
 - Register the operator for admission webhooks.
2. After applying the operator manifests, wait for the cluster to converge.

```
kubectl -n contrast-agent-operator wait  
pod --for=condition=ready --selector=app.kubernetes.io/  
name=operator,app.kubernetes.io/part-of=contrast-agent-operator --  
timeout=30s
```

```
oc -n contrast-agent-operator wait  
pod --for=condition=ready --selector=app.kubernetes.io/  
name=operator,app.kubernetes.io/part-of=contrast-agent-operator --  
timeout=30s
```

- When the wait command succeeds the operator is ready to be [configured \(page 560\)](#).

Upgrades

If the Agent Operator requires an upgrade [follow the steps here \(page 564\)](#) to complete the upgrade process.

Agent Operator configuration

The topic describes the schema for every configuration entity type the Contrast Agent Operator accepts. Some entities are optional.

AgentConfiguration

```
apiVersion: agents.contrastsecurity.com/v1beta1  
kind: AgentConfiguration  
metadata:  
  name: example-agent-configuration  
  namespace: default  
spec:  
  yaml: |  
    server:  
      environment: QA  
  suppressDefaultServerName: false  
  suppressDefaultApplicationName: false
```

Property	Type	Required	Default value	Description
spec.yaml	string	No		A YAML configuration file as documented "YAML configuration"
spec.suppressDefaultServerName	boolean	No	False	If false, automatically set the Contrast server name on injected workloads ('kubernetes-{namespace}'), rather than use the default (normally the pod name).
spec.suppressDefaultApplicationName	boolean	No	False	If false, automatically set the Contrast application name on injected workloads (the workload name), rather than use the default (generated by the agent).



NOTE

Connection keys will be ignored in the provided YAML file and should not be provided.

AgentConnection

```
apiVersion: agents.contrastsecurity.com/v1beta1  
kind: AgentConnection
```

```
metadata:
  name: example-agent-connection
  namespace: default
spec:
  url: https://app.contrastsecurity.com/Contrast
  apiKey:
    secretName: example-agent-connection-secret
    secretKey: apiKey
  serviceKey:
    secretName: example-agent-connection-secret
    secretKey: serviceKey
  userName:
    secretName: example-agent-connection-secret
    secretKey: userName
```

Property	Type	Required	Default value	Description
spec.url	string	Yes		The URL of your Contrast server.
spec.apiKey.secretName	string	Yes		The name of the Secret containing the apiKey.
spec.apiKey.secretKey	string	Yes		The key of the value in the named Secret containing the apiKey.
spec.serviceKey.secretName	string	Yes		The name of the Secret containing the serviceKey.
spec.serviceKey.secretKey	string	Yes		The key of the value in the named Secret containing the serviceKey.
spec.userName.secretName	string	Yes		The name of the Secret containing the userName.
spec.userName.secretKey	string	Yes		The key of the value in the named Secret containing the userName.



IMPORTANT

For security, Secrets referenced must be contained in the same namespace as the AgentConnection.

AgentInjector

```
apiVersion: agents.contrastsecurity.com/v1beta1
kind: AgentInjector
metadata:
  name: example-injector-dotnet-core
  namespace: default
spec:
  enabled: true
  version: latest
  type: dotnet-core
  image:
    registry: docker.io/contrast
    name: agent-dotnet-core
    pullSecretName: contrastdotnet-pull-secret
    pullPolicy: Always
  selector:
    images:
      - "*"
  labels:
```

```

- name: app
  value: example-*
connection:
  name: example-agent-connection
configuration:
  name: example-agent-configuration

```

Property	Type	Required	Default value	Description
spec.enabled	boolean	No	TRUE	Enables or disables this agent injector.
spec.version	string	No	latest	The version of the agent to inject. The literal 'latest' will inject the latest version. Partial version matches are supported, e.g. '2' will select version '2.1.0'.
spec.type	agentType	Yes		The type of agent to inject. Can be one of ['dotnet-core', 'java', 'nodejs' or 'nodejs-esm', 'php', 'python'].
spec.image.registry	string	No	docker.io/contrast	The image registry to use for downloading agent images. This registry must be accessible by the pods being injected and by the operator.
spec.image.name	string	No	{based on type}	The name of the injector image to use.
spec.image.pullSecretName	string	No		The name of a pull Secret to append to the pod's imagePullSecrets list.
spec.image.pullPolicy	string	No	Always	The pull policy to use when fetching Contrast images. See Kubernetes imagePullPolicy for more information.
spec.selector.images	string[]	No	Select all containers in Pod.	Container images to inject the agent into. Glob patterns are supported.
spec.selector.labels	labelSelector[]	No	Select all workloads in namespace.	Deployment/ StatefulSet/DaemonSet/ DeploymentConfig labels whose pods are eligible for agent injection.
spec.connection.name	string	No	Defaults AgentConnection specified by a ClusterAgentConnection.	The name of AgentConnection resource. Must exist within the same namespace.
spec.configuration.name	string	No	Defaults a AgentConfiguration specified by a ClusterAgentConfiguration.	The name of a AgentConfiguration resource. Must exist within the same namespace.

- Disabling an existing AgentInjector will remove all injections from selected workloads.
- The referenced AgentConnection and AgentConfiguration must exist in the same namespace as the AgentInjector.
- If using a custom registry, both the Pod being injected and the operator must have access, either through the default pull secret, or custom pull secrets.
- Agent version `latest` is recommended when using the agent in pre-production environments.
- The AgentInjector supports selecting Deployment, StatefulSet, DaemonSet, and DeploymentConfig (on OpenShift) workloads. Injecting pods directly is not supported.

- If the selected workload creates many containers in a single Pod, `spec.selector.images` can be used to filter which containers are injected.

labelSelector

Property	Type	Required	Default value	Description
name	string	Yes		The name of the label to match.
value	string	Yes		The value of the label to match. Glob patterns are supported.



NOTE

Label selections are cumulative using the logical AND operation.

agentType

Agent	Agent Type
.NET Core	dotnet-core
Java	java
Node.js	nodejs
	OR
	nodejs-esm
PHP	php
Python	python

Types are further documented in [Operator supported technologies \(page 553\)](#).

ClusterAgentConfiguration

```
apiVersion: agents.contrastsecurity.com/v1beta1
kind: ClusterAgentConfiguration
metadata:
  name: default-agent-configuration
  namespace: contrast-agent-operator
spec:
  namespaces:
    - default
  template:
    spec:
      yaml: |
        server:
          environment: QA
```

Property	Type	Required	Default value	Description
spec.namespace	string[]	No	All namespaces.	The namespaces to apply this AgentConfiguration template to. Glob syntax is supported.
spec.template	AgentConfiguration	Yes		The default AgentConfiguration to apply to the namespaces selected by 'spec.namespaces'.

**NOTE**

For security, ClusterAgentConfiguration manifests must be deployed into the same namespace of the operator.

ClusterAgentConnection

```
apiVersion: agents.contrastsecurity.com/v1beta1
kind: ClusterAgentConnection
metadata:
  name: default-agent-connection
  namespace: contrast-agent-operator
spec:
  namespaces:
    - default
  template:
    spec:
      url: http://app.contrastsecurity.com/Contrast
      apiKey:
        secretName: default-agent-connection-secret
        secretKey: apiKey
      serviceKey:
        secretName: default-agent-connection-secret
        secretKey: serviceKey
      userName:
        secretName: default-agent-connection-secret
        secretKey: userName
```

Property	Type	Required	Default value	Description
spec.namespace	string[]	No	All namespaces.	The namespaces to apply this AgentConfiguration template to. Glob syntax is supported.
spec.template	AgentConnection	Yes		The default AgentConnection to apply to the namespaces selected by 'spec.namespaces'.

**NOTE**

- For security, ClusterAgentConnection manifests must be deployed into the same namespace of the operator.
- Secrets referenced by ClusterAgentConnection must exist in the same namespace in which the ClusterAgentConnection entity is deployed.

See also

- [Install Agent Operator \(page 554\)](#)
- [Agent Operator supported technologies \(page 553\)](#)
- [Agent operator telemetry \(page 572\)](#)

Upgrade the operator

The Contrast Agent Operator follows [semantic versioning](#).

- MAJOR versions may include breaking changes to the operator API. Care should be taken when upgrading between MAJOR versions as manifests may have changed or existing CRDs may need to be updated.
- MINOR versions contain new features and are fully backwards compatible and are safe to apply to an existing cluster. Optional manifest changes may be needed to use new functionality.
- Patch versions contain security and bug fixes and are fully backwards compatible and are safe to apply to an existing cluster. No manifest changes are required.

Contrast publishes image tags in the following format:

```
:2
:2.1
:2.1.10
:latest
```

Where `:2` represents the latest release in the 2.X.X semantic version branch. To simplify upgrades, prefix versions may be used based on your risk tolerance (ensure `imagePullPolicy` is set to `Always`).



NOTE

While the Contrast Agent Operator supports high availability setups using multiple replicas and leader leases, Contrast only supports deployments where all operator instances are running the same version for extended periods of time. The option `imagePullPolicy` should not be relied on to keep multiple instances on the same version. Using an operator, such as [Keel](#) to facilitate safe upgrades, is recommended if automatic upgrades are desired.

Minor and patch upgrades

Upgrading to new versions follows the same steps as installing into a fresh cluster. Executing as a cluster administrator, apply the operator manifests using `kubectl` (Kubernetes) or `oc` (OpenShift).

```
kubectl apply -f https://github.com/Contrast-Security-OSS/agent-operator/releases/latest/download/install-prod.yaml
```

```
oc apply -f https://github.com/Contrast-Security-OSS/agent-operator/releases/latest/download/install-prod.yaml
```

Major upgrades

Major upgrades may include additional manifest changes. Deleting only the `contrast-agent-operator` namespace maintains the installed CRDs (and by extension any cluster configurations).

```
kubectl delete namespace contrast-agent-operator
kubectl apply -f https://github.com/Contrast-Security-OSS/agent-operator/releases/latest/download/install-prod.yaml
```

```
oc delete project contrast-agent-operator
oc apply -f https://github.com/Contrast-Security-OSS/agent-operator/releases/latest/download/install-prod.yaml
```

While these generic steps may work in most major upgrades, care should be taken to follow the migration steps provided in the release notes, if any, to ensure the major upgrade is successful.

See also

- [Agent Operator configuration \(page 560\)](#)
- [Operator supported technologies \(page 553\)](#)

Agent Operator walkthrough

Before you begin

This topic provides a complete walk-through of installing the Contrast Agent Operator and injecting an example workload as a cluster administrator using vanilla Kubernetes. **You can use this as an alternative installation method.**

To follow this example using OpenShift, the Kubernetes commands will need to be converted to their OpenShift equivalents. All commands are expected to execute within a Bash-like terminal.

You should have a basic understanding of how Kubernetes and related software work. You may need to adjust the instructions to meet your specific circumstances.

Step 1: Install the operator

To install the operator, the operator manifests must be applied to the cluster. See [Install Agent Operator with manifest files \(page 559\)](#) for installation steps. Contrast provides a single-file installation YAML that can be directly applied to a cluster and provides reasonable defaults. Additional modifications may be desired based on your specific circumstances, in which case, a configuration management framework, such as [Kustomize](#), is recommended.



NOTE

This single-file installation YAML will create and install into the `contrast-agent-operator` namespace. This namespace will be used later.

After waiting for cluster convergence, the operator should be ready in the `Running` status.

```
% kubectl -n contrast-agent-operator get pods
```

Output:

NAME	READY	STATUS	RESTARTS	AGE
contrast-agent-operator-57f5cfbf7-9svtt	1/1	Running	0	27s
contrast-agent-operator-57f5cfbf7-fp4vp	1/1	Running	0	39s

The operator is ready to be configured.

Step 2: Configure the operator

The operator must first be configured before injecting cluster workloads.

Kubernetes secrets are used to store connection authentication keys. Note that the name of the Secret created in the next part is `default-agent-connection-secret` and is created in the `contrast-agent-operator` namespace.

```
% kubectl -n contrast-agent-operator \
    create secret generic default-agent-connection-secret \
    --from-literal=apiKey=TODO \
    --from-literal=serviceKey=TODO \
    --from-literal=username=TODO
```

Output:

```
secret/default-agent-connection-secret created
```

**NOTE**

Replace TODO with the equivalent values for your Contrast server instance. [Find the agent keys \(page 104\)](#) describes how to retrieve agent keys from the Contrast UI.

To complete the connection configuration, a ClusterAgentConnection is needed. Note that ClusterAgentConnection created in the next part is created in the `contrast-agent-operator` namespace and refers to the Secret's key values used above.

```
% kubectl apply -f - <<EOF
apiVersion: agents.contrastsecurity.com/v1beta1
kind: ClusterAgentConnection
metadata:
  name: default-agent-connection
  namespace: contrast-agent-operator
spec:
  template:
    spec:
      url: https://app.contrastsecurity.com/Contrast
      apiKey:
        secretName: default-agent-connection-secret
        secretKey: apiKey
      serviceKey:
        secretName: default-agent-connection-secret
        secretKey: serviceKey
      userName:
        secretName: default-agent-connection-secret
        secretKey: userName
EOF
```

Output:

```
clusteragentconnection.agents.contrastsecurity.com/default-agent-connection
created
```

**NOTE**

The name of the ClusterAgentConnection is not important and can be named anything.

The operator is now configured and can inject agents into existing workloads.

Step 3: Inject workloads

This example will focus on injecting the Contrast Java agent into the [Java sample application](#) using a Deployment workload.

First, deploy the sample application to the cluster. Note that the Deployment created in the next part is created in the `default` namespace.

```
% kubectl apply -f - <<EOF
apiVersion: apps/v1
kind: Deployment
metadata:
  name: spring-petclinic
  namespace: default
  labels:
    arbitrary-label: arbitrary-value
spec:
  selector:
    matchLabels:
      app: spring-petclinic
  template:
    metadata:
      labels:
        app: spring-petclinic
    spec:
      containers:
        - image: contrastsecuritydemo/spring-petclinic:1.5.1
          name: spring-petclinic
EOF
```

Output:

```
deployment.apps/spring-petclinic created
```

After waiting for cluster convergence, the deployed workload should be ready in the `Running` status.

```
% kubectl -n default get pods
```

Output:

NAME	READY	STATUS	RESTARTS	AGE
spring-petclinic-77d97bdbd5-ts2cz	1/1	Running	0	15d

Next, the operator can be configured to inject the Java agent using an `AgentInjector` configuration entity. Note that the `AgentInjector` needs to be created in the same namespace that the previous Deployment was deployed into, `default` in this case.

```
% kubectl apply -f - <<EOF
apiVersion: agents.contrastsecurity.com/v1beta1
kind: AgentInjector
metadata:
  name: spring-petclinic-injector
  namespace: default
spec:
  type: java
  selector:
    labels:
      - name: arbitrary-label
        value: arbitrary-value
EOF
```

Output:

Checking the logs of the `spring-petclinic-app` Pod shows that the Contrast Java agent is now instrumenting the application.

```
% kubectl -n default logs Deployment/spring-petclinic
Defaulted container "spring-petclinic" out of: spring-petclinic, contrast-
init (init)
Picked up JAVA_TOOL_OPTIONS: -javaagent:/opt/contrast/contrast-agent.jar
[Contrast] Wed Dec 20 21:47:23 GMT 2023 Loading pre-packaged configuration
[Contrast] Wed Dec 20 21:47:23 GMT 2023 Couldn't find pre-packaged
configuration.
[Contrast] Wed Dec 20 21:47:23 GMT 2023 Starting Contrast (build 6.1.1)
Pat. 8,458,789 B2
[Contrast] Wed Dec 20 21:47:24 GMT 2023 Contrast logger configuration
errors will be logged to stderr
[Contrast] Wed Dec 20 21:47:26 GMT 2023 Copyright: 2023 Contrast Security,
Inc
[Contrast] Wed Dec 20 21:47:26 GMT 2023 Contact:
support@contrastsecurity.com
[Contrast] Wed Dec 20 21:47:26 GMT 2023 License: Commercial
[Contrast] Wed Dec 20 21:47:26 GMT 2023 NOTICE: This Software and the
patented inventions embodied within may only be used as part of
[Contrast] Wed Dec 20 21:47:26 GMT 2023 Contrast Security's commercial
offerings. Even though it is made available through public
[Contrast] Wed Dec 20 21:47:26 GMT 2023 repositories, use of this Software
is subject to the applicable End User Licensing Agreement
[Contrast] Wed Dec 20 21:47:26 GMT 2023 found at https://
www.contrastsecurity.com/enduser-terms-0317a or as otherwise agreed between
[Contrast] Wed Dec 20 21:47:26 GMT 2023 Contrast Security and the End User.
The Software may not be reverse engineered, modified,
[Contrast] Wed Dec 20 21:47:26 GMT 2023 repackaged, sold, redistributed or
otherwise used in a way not consistent with the End User
[Contrast] Wed Dec 20 21:47:26 GMT 2023 License Agreement.
[Contrast] Wed Dec 20 21:47:26 GMT 2023 The Contrast Java agent collects
usage data in order to help us improve compatibility and security coverage.
[Contrast] Wed Dec 20 21:47:26 GMT 2023 The data is anonymous and does not
contain application data. It is collected by Contrast and is never shared.
[Contrast] Wed Dec 20 21:47:26 GMT 2023 You can opt-out of telemetry by
setting the CONTRAST_AGENT_TELEMETRY_OPTOUT environment variable to 'true'
or '1'
[Contrast] Wed Dec 20 21:47:26 GMT 2023 Read more about Contrast Java agent
telemetry: https://docs.contrastsecurity.com/en/java-telemetry.html
[Contrast] Wed Dec 20 21:47:27 GMT 2023 Effective instructions:
Assess=true, Protect=false, Observe=false
[Contrast] Wed Dec 20 21:47:27 GMT 2023 Contrast logger configuration
errors will be logged to stderr
[Contrast] Wed Dec 20 21:47:41 GMT 2023 Starting JVM [18888ms]
```

[illegible]

Step 4: Uninstall the operator (optional)

To restore the original state of the cluster, first remove existing AgentInjectors.

```
% kubectl -n default delete agentinjector spring-petclinic-injector
```

Output:

```
agentinjector.agents.contrastsecurity.com "spring-petclinic-injector"
deleted
```

After which, the operator will restore all injected workloads to their previous non-instrumented state. Once the cluster converges, the operator can be safely removed.

```
% kubectl delete -f https://github.com/Contrast-Security-OSS/agent-operator/releases/latest/download/install-prod.yaml
```

Output:

```
namespace "contrast-agent-operator" deleted
customresourcedefinition.apiextensions.k8s.io
"agentconfigurations.agents.contrastsecurity.com" deleted
customresourcedefinition.apiextensions.k8s.io
"agentconnections.agents.contrastsecurity.com" deleted
customresourcedefinition.apiextensions.k8s.io
"agentinjectors.agents.contrastsecurity.com" deleted
customresourcedefinition.apiextensions.k8s.io
"clusteragentconfigurations.agents.contrastsecurity.com" deleted
customresourcedefinition.apiextensions.k8s.io
"clusteragentconnections.agents.contrastsecurity.com" deleted
serviceaccount "contrast-agent-operator-service-account" deleted
clusterrole.rbac.authorization.k8s.io "contrast-agent-operator-service-
role" deleted
clusterrolebinding.rbac.authorization.k8s.io "contrast-agent-operator-
service-role-binding" deleted
service "contrast-agent-operator" deleted
deployment.apps "contrast-agent-operator" deleted
poddisruptionbudget.policy "contrast-agent-operator" deleted
mutatingwebhookconfiguration.admissionregistration.k8s.io "contrast-web-
hook-configuration" deleted
```

See also

- [Install the Agent Operator \(page 554\)](#)
- [Agent Operator configuration \(page 560\)](#)

Uninstall the Agent Operator

The Contrast Agent Operator stores all data in the Kubernetes backplane, and is designed to completely remove all modifications when removed from a cluster. To ensure that everything is cleaned up correctly, it is recommended the following steps are taken in order.

```
kubect1 delete crd agentconfigurations.agents.contrastsecurity.com
kubect1 delete crd agentconnections.agents.contrastsecurity.com
kubect1 delete crd agentinjectors.agents.contrastsecurity.com
kubect1 delete crd clusteragentconfigurations.agents.contrastsecurity.com
kubect1 delete crd clusteragentconnections.agents.contrastsecurity.com
```

```
oc delete crd agentconfigurations.agents.contrastsecurity.com
oc delete crd agentconnections.agents.contrastsecurity.com
oc delete crd agentinjectors.agents.contrastsecurity.com
oc delete crd clusteragentconfigurations.agents.contrastsecurity.com
oc delete crd clusteragentconnections.agents.contrastsecurity.com
```

Deleting the CRDs will delete any operator configuration entities automatically. Allow the Contrast Agent Operator to reverse any changes it has made to cluster workloads once the configuration entities have been removed.

**NOTE**

This may cause substantial shifting of deployed pods as Kubernetes redeploys impacted workloads, depending on how many workloads were injected by the operator. Caution is advised in larger clusters.

After the cluster settles, the operator is safe to remove.

```
kubect1 delete -f https://github.com/Contrast-Security-OSS/agent-operator/
releases/latest/download/install-prod.yaml
```

```
oc delete -f https://github.com/Contrast-Security-OSS/agent-operator/
releases/latest/download/install-prod.yaml
```

**NOTE**

Errors around missing CRDs is normal if the CRDs were deleted in the first step as recommended.

.NET Core chaining support

The .NET Core Agent, paired with the [Contrast Agent Operator \(page 552\)](#), supports [profiler chaining with Dynatrace](#) using the [Dynatrace Operator](#). Support is enabled automatically when the Dynatrace Operator is used to inject the Dynatrace agent into workloads.

If you use the Dynatrace Operator in `classicFullStack` mode, set the `CONTRAST_ENABLE_EARLY_CHAINING=true` environment variable for the Contrast Agent Operator and restart the affected pods. Use this [manifest example](#) to guide you.

The Agent Operator supports executing in OpenShift clusters that have security context constraints (SCCs) using the built-in admission or mutating controllers. The Dynatrace Operator and any injections are tested against the `restricted` policy. If you are running your application in an OpenShift cluster where the default restricted policy disallows setting an SCCs policy, ensure the `CONTRAST_SUPPRESS_SECCOMP_PROFILE=true` environment variable is applied to the Agent Operator workload.

Vendor	Version	Support validated on
Dynatrace	Operator v0.6.0	2022/06/09

Future Dynatrace versions may break chaining. Chaining can introduce incompatibilities and can be disabled using the `agent.dotnet.enable_chaining: false` option.

Agent Operator Telemetry

The Contrast Agent Operator uses telemetry to collect usage data. Telemetry is collected when the operator is first installed in a cluster and then periodically (every few hours) afterwards.

Your privacy is important to us. The telemetry feature doesn't collect application data. The data is anonymized before being sent securely to Contrast. Then the aggregated data is stored encrypted and under restricted access control. Any collected data will be deleted after one year.

To opt-out of the telemetry feature, set the `CONTRAST_AGENT_TELEMETRY_OPTOUT` environment variable to `1` or `true`.

Telemetry data is securely sent to *telemetry.dotnet.contrastsecurity.com*. You can also opt out of telemetry by blocking communication at the network level.

The telemetry feature collects the following data:

Operator v0.3.0

- The version of the operator.
- The uptime of the operator.
- Cluster version information and platform as published by the Kubernetes API.
- A cryptographically (SHA256) anonymous hash of the cluster ID, a randomly generated GUID created at first launch by the operator and stored in the operator's namespace as a Secret.
- The count of watched resources in the cluster (all operator entities, DaemonSets, DeploymentConfigs, Deployments, Namespaces, Pods, Secrets, and StatefulSets).
- Exceptions thrown internally by the operator, including log message, exception type, exception message, and stack trace frames.

Pod restart

Kubernetes pods are designed to maintain a desired state defined in YAML configurations. When changes occur to configurations that affect a pod's runtime environment, Kubernetes initiates pod restarts to ensure the actual state aligns with the desired state. This reconciliation process is fundamental to Kubernetes' operation.

Specifically, modifications to resources such as `AgentConfiguration`, `AgentConnection`, and `AgentInjector` trigger these restarts. These resources define critical aspects of the pod's behavior, including connectivity, settings, and injection rules. When these configurations are updated, Kubernetes detects the discrepancy between the current pod state and the new desired state.

The impact extends to `ClusterAgentConnection` and `ClusterAgentConfiguration` too. These cluster-level resources generate `AgentConnections` and `AgentConfigurations` within specified namespaces. Therefore, changes to these cluster-level resources cascade down, leading to corresponding updates and restarts of pods in the affected namespaces.

Any change to the pod's configuration, whether direct or indirect through related resources, necessitates a restart to guarantee that the pod reflects the latest defined state within the cluster. This ensures consistency and reliability across the application. It is essential to understand that only the pods affected explicitly by the configuration change will be restarted. For instance, if an `AgentInjector` targets deployments using a specific label selector, only the pods within those targeted deployments will be restarted, not all pods in the cluster.

Agent performance

Agent performance in application security refers to the effectiveness and efficiency of security agents or software tools deployed to protect applications from vulnerabilities and threats. These agents can be various security mechanisms such as firewalls, intrusion detection systems (IDS), intrusion prevention systems (IPS), web application firewalls (WAF), and vulnerability scanners.

See also

[Agent performance with Protect \(page 573\)](#)

[Java agent performance \(page 574\)](#)

[.NET Core agent performance \(page 575\)](#)

Agent performance with Protect

What can impact Protect performance?

The performance of these agents is crucial because they need to accurately and swiftly detect and respond to potential security risks in real-time. Here are a few main aspects related to agent performance in application security:

1. **Accuracy:** The security agent should have a high level of accuracy in identifying and classifying security threats. To address genuine risks, we want to minimize false positives (misidentifying legitimate activities as threats) and false negatives (failing to detect actual threats).
 - When sensitive rules are detected, they are audited and validated. The rules are adjusted or turned off for a quick fix. The engineering team is responsible for fixing and tuning the false positives (FP) to ensure the system's accuracy.
2. **Speed and responsiveness:** Application security agents should operate in real-time or near real-time to detect and respond promptly to security incidents. Delayed responses can lead to extended exposure to vulnerabilities and increase the risk of successful attacks.
 - Contrast's infrastructure services are expected to have enough resources and capacity to handle high loads and errors without impacting performance.
3. **Scalability:** Agents should be capable of handling the demands of large-scale applications and networks. The system has a cloud-native licensing model that allows easy deployment and scaling of microservices in clustered environments. It provides flexibility for scaling up and down based on demand to ensure optimal performance.
 - Our updated configuration deployment enables the efficient scaling up and down of the system with microservices. It simplifies the process of deploying and managing the system at scale, ensuring smooth operations.
4. **Resource utilization:** Effective security agents should need to utilize system resources, such as CPU, memory, and network bandwidth, efficiently. They should handle the need for robust security measures and minimize resource consumption to avoid adversely impacting the performance of the protected applications.
 - The memory consumption is typically capped. See the agent-specific sections for detailed information about memory usage.
5. **Adaptability:** Application security agents should be adaptable to evolving threats and new attack vectors. Regular updates and enhancements to the agent's capabilities, such as signature updates, rule updates, and machine learning models, are necessary to ensure the continued effectiveness of security measures.
 - You can optimize performance by determining the most applicable rules and tuning them. You can also use exclusions and allowlisting to reduce unnecessary scans and analysis, which helps in further tuning and reducing performance impact.

Overall, the agent performance objective in application security is to provide a robust and reliable defence against potential threats while minimizing false positives, response times, and resource

utilization. Organizations can enhance the security posture of their applications and protect them from various attacks by balancing security effectiveness and system performance.

See also

[Java agent performance \(page 574\)](#)

[.NET Core agent performance \(page 575\)](#)

Performance expectations for the Java agent with Protect

To provide insights into its impact on performance, we have gathered information based on internal tests conducted with a sample Java application. Please note that the actual performance numbers may vary based on several other factors.

CPU Usage

- The Java agent typically utilizes 5-15% of CPU resources
- This range provides an estimate of the additional CPU load incurred when the agent is enabled in Protect mode
- The impact may vary based on the complexity of your application, the load it experiences, and other environmental factors

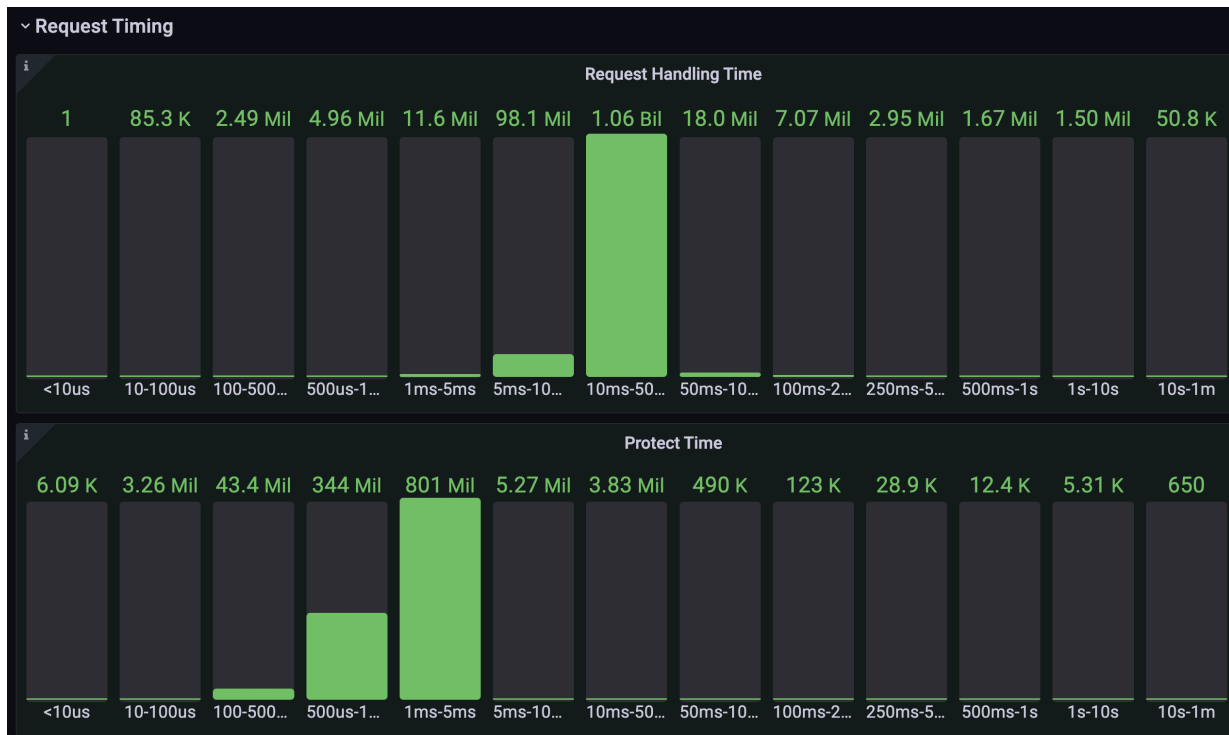
Memory consumption

- The Java agent typically requires 200-300 MB of additional memory
- This memory usage accounts for the Agent's operations and any associated data structures
- The actual memory impact may depend on the size and complexity of your application, as well as the number of concurrent requests it handles

Latency

- Enabling the Java agent in Protect mode may introduce additional latency to the processing of requests
- 99% of observed requests for Protect analysis take between 100 μ s - 5 ms
- The specific impact will depend on various factors like the application's workload, the number of concurrent users, and the nature of the requests
- It is important to note that the latency introduced is generally minimal, but it may vary based on your specific application characteristics. These performance numbers serve as a baseline to provide you with an initial understanding of the Java agent's impact on your application. However, we emphasize that the actual performance impact may differ based on the unique aspects of your environment and application. We recommend conducting your performance tests using representative workloads and realistic scenarios to accurately gauge the impact on your application's performance. By simulating conditions similar to your production environment, you can fine-tune the configuration and evaluate the trade-off between security and performance. Our team is committed to continuously enhancing the Java agent's efficiency and optimizing its impact on your application's performance. We value your feedback and encourage you to reach out to us with any performance-related observations or concerns. Feel free to [contact our support team](#) if you require any assistance or further information regarding the performance characteristics of our Java agent. We are here to help you ensure the security and stability of your application while maintaining optimal performance levels.

The following graph illustrates a combined representation of the request handling time for real customer host applications instrumented with our agent. It showcases a 24-hour sample of Java performance impact.



- The top row shows the overall request handling time, indicating that a substantial portion of the HTTP requests, approximately 93% of requests (1.06 billion hits), fall within the 10 millisecond to 50 milliseconds range. Additionally, you can see 98.1 million hits with a handling time between 5 and 10 milliseconds, while the remaining handling times follow a normal distribution.
- The bottom row displays the Protect time, which represents the length of time our Protect RASP product takes to analyze each request. Notably, among the total requests, 800 million were analyzed within a time frame of 1 to 5 milliseconds, demonstrating a negligible impact on latency. Similarly, an additional 344 million requests underwent analysis within an even shorter time range of 0.5 to 1 milliseconds, indicating minimal latency impact. The distribution of request handling times for host applications instrumented with our agent allows us to observe the number of requests falling into different time ranges while also providing insights on the performance impact introduced by Protect.
- Garbage collecting and other JVM elements can pause execution, which impacts timing.

Performance expectations for the .NET Core agent with Protect

To provide insights into its impact on performance, we have gathered information based on internal tests conducted with a sample .NET Core application. Please note the actual performance numbers may vary based on several other factors.

CPU usage

- The .NET Core agent typically utilizes 5-10% of CPU resources
- This range represents the additional CPU load incurred when the agent is enabled in Protect mode
- The actual impact may vary depending on the complexity of your application, the workload it handles, and other environmental considerations

Memory consumption

- The .NET Core agent typically requires around 200 MB of additional memory
- This memory usage accounts for the agent's operations and relevant data structures
- The actual memory impact may differ based on your application's size, complexity, and the number of concurrent requests it processes

Latency

- Enabling the .NET Core agent in Protect mode may introduce additional latency during request processing
- The latency increase can range from 1-10 milliseconds
- The specific impact on latency depends on factors such as your application's workload, the number of concurrent users, and the nature of the requests being handled
- It is important to note that the introduced latency is generally minimal, but actual numbers may vary based on your specific application characteristics. These performance numbers serve as a baseline, providing an initial understanding of the impact the .NET Core agent has on your application's performance. However, it is crucial to recognize that the actual performance impact may differ based on the unique aspects of your environment and application. We strongly recommend conducting your performance tests using representative workloads and realistic scenarios to accurately assess the impact on your application's performance. By simulating conditions similar to your production environment, you can optimize the configuration and evaluate the balance between security and performance. Our team remains dedicated to continuously improving the efficiency of the .NET Core agent and optimizing its impact on your application's performance. We value your feedback and encourage you to reach out to us with any performance-related observations or concerns. Feel free to [contact our support team](#) if you require any assistance or further information regarding the performance characteristics of our .NET Core agent. We are here to assist you in ensuring the security and stability of your application while maintaining optimal performance levels.

Incident management with ADR

Application Detection and Response (ADR) is a powerful application-layer security solution that is designed specifically to monitor, detect, and prevent application layer attacks in production environments.

It also provides comprehensive visibility into the application layer attack surface and insights into vulnerabilities that remove blind spots for defenders and enable faster, more accurate detection and response.

ADR Benefits

Contrast ADR defends your applications with:

- **Zero-day protection:** Runtime protection blocking for known and unknown vulnerabilities.
- **Real-time monitoring:** Detects and alerts on anomalous behavior within the application layer.
- **Actionable Alerts:** Know the context for all application alerts through a summary of suspicious activity, payloads, Indicators of Compromise (IoC), and more.
- **Runtime observability:** Real-time security blueprints provide context to incidents better to assess the severity and impact of an attack.
- **Guided runbooks:** Clear, actionable steps to quickly identify true positive attacks and contain threats.

- **SIEM integration:**

You can ingest ADR alerts, events, attack payload, and vulnerability data into your SIEM tools for effective monitoring and triage.

Contrast currently supports these SIEM integrations:

- Azure Sentinel
- Data Dog
- Splunk
- Sumo logic

How ADR works

1. **Integrated agent:** Integrate a Contrast agent into your application code.
2. **Policies:** Set policies for rules that monitor or block threats while your application is in use.
3. **Monitor and protect:** Observe attack events and adjust policies, as needed.
4. **Mitigation actions:** Review the suggestions that Contrast provides for reducing threats to your applications based on a set of rules.

Requirements for ADR

To use ADR, make sure Contrast Protect is turned on and the server environment is set to **DEVELOPMENT**, **QA**, or **PRODUCTION**.

[Enable Protect \(page 1155\)](#) describes how to turn on Protect.

See also

- [Attack events \(hosted customers\) \(page 578\)](#)
- [Attacks \(page 584\)](#)
- [Vulnerabilities \(page 1021\) \(IAST\)](#)
- [Libraries \(page 922\) \(SCA\)](#)

- [Contrast Security observability \(page 606\)](#)

Vulnerability assessment with AVM

Contrast Security combines the power of Application Detection and Response (ADR) with an optional Application Vulnerability Monitoring (AVM) add-on to address critical application security challenges.

Benefits

The combination of ADR and AVM helps organizations to reduce risk by providing continuous visibility into attacks and vulnerabilities in production environments. AVM goes beyond static code analysis, theoretical assessments, and simulated testing to pinpoint the weaknesses that actually pose the greatest risk in production environments.

Features

- **Continuous monitoring:** Real-time visibility into exploitable vulnerabilities
- **Risk insight:** Vulnerability awareness with severity
- **Attack context:** Connect vulnerabilities to attacks to guide triage prioritization
- **Security controls:** Implement ADR security controls to mitigate risk while remediation is in progress
- **Collaboration:** Notify development team about real risk that is live in production to help prioritize remediation
- **Reports:** AVM vulnerability reports let security teams help AppSec to prioritize remediation efforts and also respond to threats effectively.

To get the most out of your AVM solution, visit our [vulnerability \(page 1021\)](#) topics.

Requirements

To use AVM, make sure Contrast Assess is turned on and the server environment is set to **PRODUCTION**.

[Enable Assess \(page 1153\)](#) describes how to turn on Assess.

Attack events ☹️



NOTE

The Attacks events view is for hosted customers.

If you are an on-premises customer, visit [Attacks \(page 584\)](#).

Attack events are any actions that unauthorized individuals or groups take to damage, disrupt, or gain illegal access to an application's systems, data, or functionality. Examples of attack events include:

- **SQL injection:** An attacker inserts malicious code into your application's database queries to steal data or take control of your systems.
- **Cross-site scripting (XSS):** An attacker injects malicious scripts into your application to steal user data or redirect them to malicious websites.
- **API attacks:** An attacker exploits vulnerabilities in your APIs to gain unauthorized access to data or functionality.

Event data retention

Contrast keeps attack event data for up to a year. You can also:

- [Output to syslog \(page 918\)](#)
- [Set up a generic webhook \(page 1076\)](#)
A webhook receives data in a POST request only when a specified event occurs. When the webhook sees the event, it collects the data and sends it to the specified URL.
- [Integrate ADR \(page 1051\)](#) with Security Information and Event Management (SIEM) tools

Exclusion of PROBED event data

You can choose to stop saving data for attack events with a **PROBED** result. You might want to do this to improve performance in the Contrast web interface.

To stop saving PROBED event data, contact [Customer Support Customer Support](#).

Tasks

In Contrast you can:

- [View attack events \(hosted customers\) \(page 579\)](#)
- [Manage attack events \(hosted customers\) \(page 583\)](#)

View attack events ☹



NOTE

This feature is for hosted customers.

If you are an on-premises customer, visit [View attacks \(page 585\)](#).

An [attack event](#) occurs when there is a violation of Protect rules or other suspicious application activity in instrumented applications.

Before you begin

- Ensure that [role-based access control \(page 1277\)](#) is turned on.
- If role based access control is turned on, you need a role with the View Attack Data action. Contact [Contrast support Contrast support](#) for access to this action.

Steps

1. Select **Attack events** in the header.
2. Set the main view by selecting an option in **Group by**:
 - If you want to view groups of attack events, select a group type (currently the only option is Source IP).
For example, if a source IP address of 111.111.111.111 has multiple attack events, grouping by Source IP displays an aggregated view for all the events.
 - Group by Source IP is the default selection.
 - If you want to view all individual attack events, clear the group by selection by moving your cursor to the Group by box and selecting the **Delete** (✕) icon.
3. To refine the view, open the filter panel by selecting **Open filters**.
Use any of these filters:
 - **Date range**: Select a date range or select **Custom** to specify a preferred data range.
The default date range is 12 hours.




- **Severity:** Select one or more vulnerability severity levels.
 - **Results:** Select one or more result types for an attack event.
 - **Rules:** Select one or more of the Protect rules associated with the attack event.
 - **Application:** Select one or more of the available applications.
 - **Environments:** Select one or more server environments.
 - **Source IP:** Select one or more source IP addresses associated with the attack event.
4. To view details about a specific attack event, select an individual attack event (not a group) which opens the Attack details view. This view includes:
- **Overview tab:** An overview of the attack event details, including these details:
 - **Source IP:** The IP address where the attack event originated.
 - **Application:** The application affected by the attack event.
 - **Server:** The server associated with the application.
 - **Rule:** The Contrast rule associated with the attack event.
 - **Severity:** The severity that Contrast assigned to the attack event.
 - **Attack value:** The suspicious value that Contrast observed.
 - **Vector analysis:** The different pathways or methods that Contrast observed where a malicious attacker could gain access to your system.
 - **Request details:** The details that Contrast observed in a request associated with the attack event.
 - **MITRE ATT&CK tactics:** Links to details for MITRE ATT&CK tactics.

The MITRE ATT&CK framework is a knowledge base of adversary tactics and techniques derived from real-world observations.

A single attack event can map to multiple tactics. In the case where multi-stage attacks events occur, an observed event might represent a single action within a larger attack chain. Alternatively, it could indicate a threat vector.

Combining event data with context from other security tools, such as Web Application Firewalls (WAFs) or Endpoint Detection and Response (EDR) solutions, allows for more precise identification of tactics. This refinement helps you to understand the full scope of an attack.

Mapping events to ATT&CK tactics is crucial for risk assessment. It enables you to identify high-risk areas and prioritize the development of new detections. This process leads to expanded security coverage.

For more information, visit [MITRE ATT&CK](#).
 - **Recommended steps of action:** Actions that Contrast recommends to protect against the attack event.
 - **Code location tab:** Where available, details about the location in your code where Contrast detected the attack event. If no information is available, this tab is not displayed. The displayed details include:
 - **File:** The file associated with the attack event.
 - **Method:** The method associated with the attack event.
 - **Stack:** The code stack associated with the attack event.
 - **Actions:** These actions are available:
 -  [Configure a Protect rule \(page 1133\)](#) to block or monitor the event
 -  [Create an exclusion \(page 1142\)](#)
 -  [Add the source IP to the denylist \(page 1149\)](#)

In the Add IP to denylist window, enter these details and select **Add**.

 - The name of the policy.
 - The IP address and, optionally, the subnet mask you want to deny. Classless Inter Domain Routing (CIDR) notation is supported for IPv4 and IPv6.

The IP source address for the selected event is displayed as the default value.

- An expiration time.
- [Create a virtual patch \(page 1138\)](#)
- [Download the event details to a JSON file](#)

SQL Injection from 192. [redacted]

Exploited 2025/28/02 05:22 PM URL: [redacted]

Overview Code Location

Source IP	Application	Server	Rule	Severity
192. [redacted]	Web-Application- [redacted]	Petclinic- [redacted]	SQL Injection	Critical

Attack Value

Contrast observed the following suspicious value accessing the application through the HTTP Request Parameter lastName

' or 1=1; # 100_staging_3

Vector Analysis

Contrast observed this value altering the meaning of the SQL query executed within org.springframework.samples.petclinic.customer.CustomerRepository.findByName(CustomerRepository.java:31) again.

SELECT DISTINCT * FROM customers WHERE customers.last_name LIKE '% ' or 1=1; # 100_staging_3%

Request Details

GET /customers?lastName=contrast-redacted-name http/1.1

accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
accept-encoding: gzip, deflate
accept-language: en-US,en;q=0.9
connection: keep-alive
Cookie: JSESSIONID [redacted]
host: localhost:8080

MITRE ATT&CK

This event may be an indicator of the following tactics:

- [Initial Access](#)
- [Execution](#)
- [Persistence](#)
- [Privilege Escalation](#)
- [Defense Evasion](#)
- [Credential Access](#)
- [Discovery](#)
- [Collection](#)
- [Command and Control](#)

Recommended Steps Of Action

- We did not block this attack because blocking was not enabled for Web-Application-bsow-adr-5 in QA. [Configure Protect Rule](#)
- Add an exclusion for this attack event. [Create Exclusion](#)

Attack event views

Depending on whether you are using a grouped view or an individual view, the Attack Events list displays these details:

Column	Grouped view	Individual view
Source IP	The IP address where multiple attack events originated. To see this detail for each attack event, select the group row or clear the Group by selection.	The IP address from which an attack event originated.

Column	Grouped view	Individual view
Severity	A severity bar that shows the number of severity types for the attack events in the group.	<p>The severity of the attack event:</p> <ul style="list-style-type: none"> • Critical: Immediate action is required. <ul style="list-style-type: none"> • The attack event impacts a large number of users, renders an application inoperable, or does irreparable damage to the organization. • Maps to this result: Exploited. • High (deprecated): Immediate action is required. <ul style="list-style-type: none"> • The event impacts a moderate number of users, substantially degrades the functioning of an application, or does substantial damage to the organization. • Maps to this result: Exploited • Medium: Action is required, but the threat is not serious. <ul style="list-style-type: none"> • The event impacts a small number of users, minimally degrades the functioning of an application, or does minor damage to the organization. • Maps to this result: Suspicious • Low: Suspicious activity. <ul style="list-style-type: none"> • May be an indicator of a larger threat. • Maps to this result: Probed • Informational: No action required. <ul style="list-style-type: none"> • Maps to this result: Blocked
Rules	<p>The number of Contrast rules that the attack value violated.</p> <p>To see the name of the rule for each attack event, select the group row or clear the Group by selection.</p>	The name of the Contrast rule that the attack value violated.
Applications	<p>The number of applications where Contrast detected the attack event.</p> <p>To see the names of the applications, select the group row or clear the Group by selection.</p>	The name of the Contrast rule that the attack value violated.
Servers	<p>The number of servers where Contrast detected the attack event.</p> <p>To see the name of the server for each attack event, select the group row or clear the Group by selection.</p>	The name of the server where Contrast detected the attack event.
Detected	<p>The time frame when Contrast detected the attack events in the group.</p> <p>To see the detected time for each attack event, select the group row or clear the Group by selection.</p>	The time when Contrast detected the attack event.

Column	Grouped view	Individual view
Result	<p>A result bar that shows the number of result types for the attack events in the group.</p> <p>To see the result for each attack event, select the group row or clear the Group by selection.</p>	<p>The result for the attack event.</p> <p>The possible results are, in order of severity:</p> <ul style="list-style-type: none"> • Exploited: <ul style="list-style-type: none"> • Contrast detected an attack event at the perimeter and confirmed it at the sink. The mode is set to Monitor. • Maps to this severity: Critical or High • Suspicious: <ul style="list-style-type: none"> • Contrast detected a low confidence attack event at the perimeter for a perimeter-only rule in Block mode. • Contrast detected a high or low confidence attack event at the perimeter for a perimeter-only rule in Monitor mode. • Contrast detected an attack event using sink-only heuristics. The mode is set to Monitor. • Maps to this severity: Medium • Blocked: <ul style="list-style-type: none"> • Contrast detected an attack event at the perimeter and confirmed it at the sink. The mode is set to Block. • Contrast detected an attack using sink-only heuristics. The mode is set to Block. • Maps to this severity: Informational • Probed: <ul style="list-style-type: none"> • Contrast detected an attack event at the perimeter, but did NOT confirm it at the sink. The mode is set to Block or Monitor. • These are ineffective attacks that can indicate an attacker is probing, scanning, or fuzzing your application for vulnerabilities. • Maps to this severity: Low
URL	<p>Not shown.</p> <p>To view the URL for each attack event, select the group row or clear the Group by selection.</p>	<p>The path the attacker used for the attack event.</p>
Attack value	<p>Not shown.</p> <p>To view the attack value for each attack event, select the group row or clear the Group by selection.</p>	<p>The value that the attacker sent that the Contrast agent detected was going to a sink.</p>
Actions	<p>The possible action you can take for an attack event in the grouped view:</p> <ul style="list-style-type: none"> • Add the IP address to a denylist <p>To view additional actions for each attack value, select the group row or clear the Group by selection.</p>	<p>The possible actions you can take for the attack event:</p> <ul style="list-style-type: none"> • Configure a Protect rule. • Create an exclusion • Add the IP address to a denylist • Download the event details to a JSON file

See also

[Manage attack events \(hosted customers\) \(page 583\)](#)

Manage attack events ☹️



NOTE

This feature is for hosted customers.

If you are an on-premises customer, visit [Manage attacks \(page 589\)](#).

When Contrast reports an attack event, you have the option of configuring a Protect rule to determine how Contrast reacts to the event or adding an application exclusion to suppress the event that you consider to be safe.

Before you begin

- Ensure that [role-based access control \(page 1277\)](#) is turned on.
Role-based access control is a preview feature and not available to all customers. Contact your Contrast representative to have role-based access control turned on.
- You need a role with the View Attack Data action.

Steps

1. Select **Attack events** in the header.
2. If you are using a grouped view, select an event row. If you are using an individual view, skip this step.
3. To configure a [Protect rule \(page 1130\)](#) for the event, select the **Configure Protect rule** icon (🛡️) in the Actions column.
Configuring a Protect rule lets you decide whether Contrast should monitor or block the event if it occurs.
4. To create an [application exclusion \(page 1142\)](#) for the event, select the **Configure exclusion** icon (🔒) in the Actions column.
Configuring an application exclusion lets you specify whether Contrast should suppress events that you know are safe.
5. To add the IP address to a [denylist \(page 1149\)](#), select the Denylist icon (🚫) in the Actions column.

Attacks ↕



NOTE

This topic is for on-premises customers.

If you are a hosted customer, visit [Attack events \(page 578\)](#).

Attacks are groups of attack events that target applications and servers. There are multiple attack events that Contrast includes in an attack, including, but not limited to:

- SQL injection
- Untrusted deserialization
- Command injection
- Many other common vulnerability types

When Contrast detects multiple attack events from the same IP address within 60 minutes, Contrast groups these events together as an attack. If Contrast sees new events from the same IP address after you fix the code, Contrast shows a new attack.

The displayed dates on the dashboard for attacks seen are based on the time when the Contrast task runs on the server, not your local timezone.

Event data retention

Contrast keeps attack event data for 30 days before removing it. To keep attack data for a longer amount of time, do the following:

- [Output to syslog \(page 918\)](#)
- [Set up a generic webhook \(page 1076\)](#)
A webhook receives data in a POST request only when a specified event occurs. When the webhook sees the event, it collects the data and sends it to the specified URL.
- Select the arrow at the end of the attack row and then select **Export attack (CSV)** or **(XML)** from the menu.

Effective (19) Find Attack 05/09/2021 03:28 pm - 06/08/2021 03:28 pm Advanced

<input type="checkbox"/> Source IP	Status	Application	Server	Rule	Start	End	Events	
<input type="checkbox"/> 127.0.0.1	BLOCKED			Command Injection	a day ago	a day ago	1	
<input type="checkbox"/> 1.2.3.5	BLOCKED			Command Injection Command Injection - Command Backdoors	5 days ago	5 days ago		<div>Denylist IP Suppress attack Export attack (CSV) Export attack (XML)</div>

Exclusion of PROBED event data

You can choose to stop saving data for attack events with a **PROBED** result. You might want to do this to reduce your database storage consumption for attacks that have a lower likelihood of being problematic.

To stop saving PROBED event data, use this JVM argument in the `contrast-server.vmoptions` file.

```
contrast.feature.TS-37298_do_not_save_probed_attack_events=on
```

This setting applies to all organization in your Contrast instance.

Tasks

In Contrast, you can:

- [View attack details \(page 585\)](#) such as which application and server was attacked and the location in the code where the attack occurred.
- [Manage attacks \(page 589\)](#) by taking actions on attacks and attack events. For example, you can configure a Protect rule for specific attack events.
- [Monitor attacks \(page 588\)](#) in an overview of current and past attacks.

View attacks ↗



NOTE

This topic is for on-premises customers.

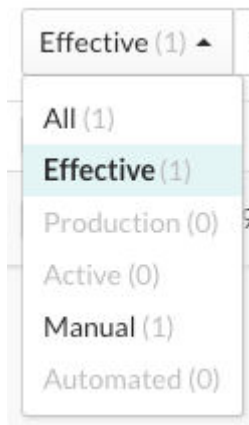
If you are a hosted customer, visit [View attack events \(page 579\)](#).

The Attacks list shows all attacks that have occurred in an organization.

Monitor Attacks Attack Events									
Effective (25) Find Attack Set Date Range Advanced									
Source IP	Status	Application	Server	Rule	Start	End	Events		
<input type="checkbox"/> 127.0.0.1	EXPLOITED	myapplication/new	myserver-local-mycompany	Command Injection - Chained Commands Command Injection - Command Backdoors	7 hours ago	7 hours ago	3		
<input type="checkbox"/> 127.0.0.1	BLOCKED	myapplication/new	myserver-local-mycompany	Path Traversal Server-Side JavaScript Injection	a day ago	a day ago	9		
<input type="checkbox"/> 127.0.0.1	BLOCKED	myapplication/new	myserver-local-mycompany	Path Traversal	a day ago	a day ago	9		
<input type="checkbox"/> 127.0.0.1	BLOCKED	myapplication/new	myserver-local-mycompany	Path Traversal	a day ago	a day ago	12		

Steps

1. Select **Attacks** in the header.
2. To view all attacks that occurred in your organization, select the **Attacks** tab.
3. To filter the view, select one of these filters:
 - **All:** Shows all attacks
 - **Effective:** Shows attacks with a status of **Blocked**, **Suspicious**, and **Exploited**.
 - **Production:** Shows attacks that occurred in a production server
 - **Active:** Shows attacks that are currently in progress
 - **Manual:** Shows attacks that have less than 20 requests per second. It could indicate that a human is generating attacks.
 - **Automated:** Shows attacks that have more than 20 requests per second. It could indicate that a malicious bot is generating attacks.



4. To view more details on the attack, select **source name** or **IP address** in the Source IP column.
5. To see each attack event in the attack, select the **Overview** tab.
6. For more filters, select **Advanced** next to the date range and select a filter.

Advanced

☐ Show suppressed

☐ Show bots blocked

☐ Show blacklisted

Tags

Attack Severity

Low

Medium

High

(0)

(9)

(0)

Results

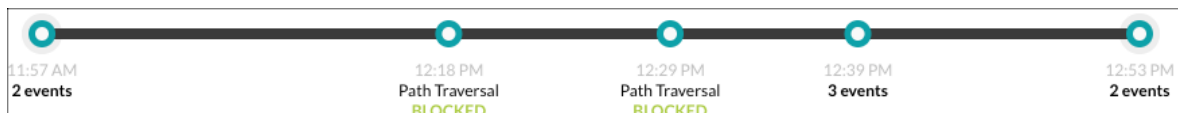
Rules

Applications

Servers

Environments

7. To see more details about the attack event, select **Source IP**.
8. To view the time of each event, under **Attack duration**, select **See timeline**.



9. To see more details including the rate of events, severity, and attacker, select the **Notes** tab.
10. To share or view communications with your team, select the **Discussion** tab.
Read existing comments or enter a new comment and click **Add Comment**.

Attack details

The attack data that Contrast displays includes these items:

- **Source IP:** The IP address from which the attack is originating.
- **Status:** The current status of the attack.
An attack status is determined by the highest severity status of the attack events within the attack. If any event has an **Exploited** status, the attack status will be **Exploited**. If there are no Exploited events, the status will be the next highest severity event's status. The severity order, starting with the highest, is:
 - **Exploited:** If Contrast detects and confirms a definite attack but the mode for the applicable rule is set to **Monitor**, Contrast does not block the request.
This status only applies to input tracing rules where Contrast is confident that an attack occurred. Confirmed attacks are those that met a high confidence threshold at the perimeter, or those that are watched and verified at the sink.

- **Suspicious:** If Contrast detects an attack and the applicable rule reports the attack as suspicious, but the mode for the rule is set to **Monitor**, Contrast does not block the request. This status applies to non-input tracing rules, where Contrast is unable to verify that an attack occurred.
- **Blocked:** If Contrast detects an attack and the mode for the applicable rule is set to **Block**, Contrast blocks the request.
- **Blocked (P):** This status applies to rules that support both Block At Perimeter and Block modes. If Contrast detects an attack before the application can process the request and the mode for the applicable rule is set to **Block At Perimeter**, Contrast blocks the request. If Contrast detects an attack that is not at the perimeter, it blocks the request and the status is **Blocked**, even if the mode for the rule is set to **Block At Perimeter**.
- **Probed:** If Contrast detects an attack but cannot confirm it and the mode for the applicable rule is set to **Monitor**, Contrast does not block the attack. Unconfirmed attacks are those that did not meet a high confidence threshold at the perimeter; they are watched but not detected at the sink.
- **Application:** Specific applications that saw attack events from the IP address while the attack was active.
- **Server:** Specific server that saw attack events from the IP address while the attack was active.
- **Rule:** Any attack type identified from the IP address while the attack was active.
- **Start:** The timestamp of the first attack event seen from the IP address during the attack time frame.
- **End:** The timestamp of the last attack event seen from the IP address during the attack time frame.
- **Events:** The number of attack events that comprise the attack.

Monitor attacks ↗



NOTE

This procedure is for on-premises customers.

You can see an overview of current and past attacks, the IP addresses of attackers, attack types, and which applications were exploited.

1. Select **Attacks** in the header.
2. If an attacker has a [source name \(page 1150\)](#), you can hover over it to see a list of associated IP addresses .
If an attacker doesn't have a source name, their avatar will show a question mark. If an attacker has successfully exploited an application, their avatar will be red.



NOTE

If the data reported for an attack event matches more than one source name, Contrast applies the most recently updated name.

Click on a IP address or source name to see details for that attacker.

3. To filter attacks you can:
 - Filter attacks by date range and environment.
 - Search to find attacks by a specific attacker IP or source name.
 - Search by affected applications or specific Assess or Protect rules.
 - Check **Show probed** to include information for attack events that resulted in a "Probed" status.
4. Under **Attack events** you can see a list of the types of attacks detected, along with the total number of attack events per type.

5. Under **Target application**, you can see each application targeted by an attack.

Manage attacks



NOTE

This procedure is for on-premises customers.

If you are a hosted customer, visit [Manage attack events. \(page 583\)](#)

Before you begin

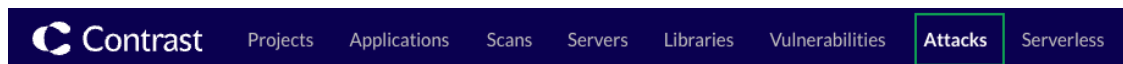
Ensure that Protect is enabled on the servers that host your applications. The Contrast header displays Attacks only when Protect is enabled.

Steps

To take action on attacks and attack events, use the following procedures.

1. View attacks or attack events.


- a. Select **Attacks** in the header.



- b. Select the **Attacks** tab or the **Attack Events** tab.


2. (Optional) Tag attacks or attack events.

Tagging attacks or attack events lets you organize them for better search results.

- a. Select one or more attacks or attack events.
- b. Select the tag icon () above the list.
- c. In the Tag Attacks window, enter a name for one or more tags.

3. Suppress attacks or events.

Suppressing attacks removes an attack and its related events from view. To suppress an attack or an attack event, use the following procedure:

- a. On the Attacks or Attacks Events list, select the check box for one or more rows and select the **Suppress Attacks** or **Suppress Events** icon ().
Alternatively, select the arrow at the end of a row and select the **Suppress Attacks** or **Suppress Events** option in the dropdown.
- b. Click **Suppress**.

4. Block IP addresses

This option [blocks a specified IP address \(page 1149\)](#). Blocking an IP address prevents unwanted activity from a specific IP address in the future.

- a. At the end of an attack or attack events row, select the dropdown ().
- b. From the menu, select **Denylist IP**.
- c. Enter a name for the rule that blocks the specified IP address.
- d. Select a date when the block expires.
- e. Click **Save**.

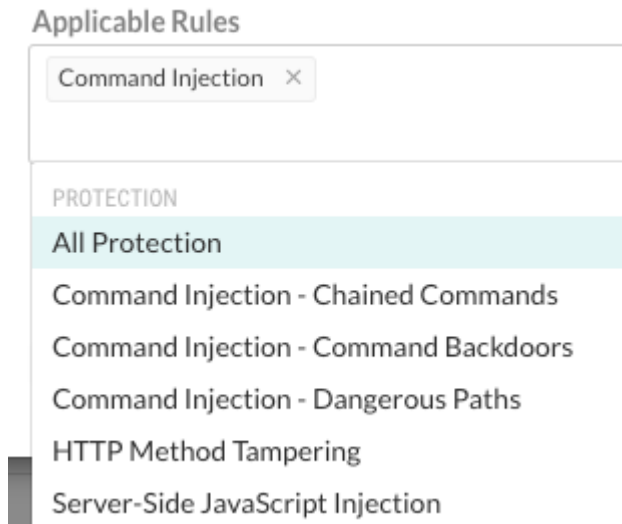
5. Add exclusions (attack events)

[Adding application exclusions \(page 1142\)](#) lets you exclude certain applications, or parts of them, from security analysis.

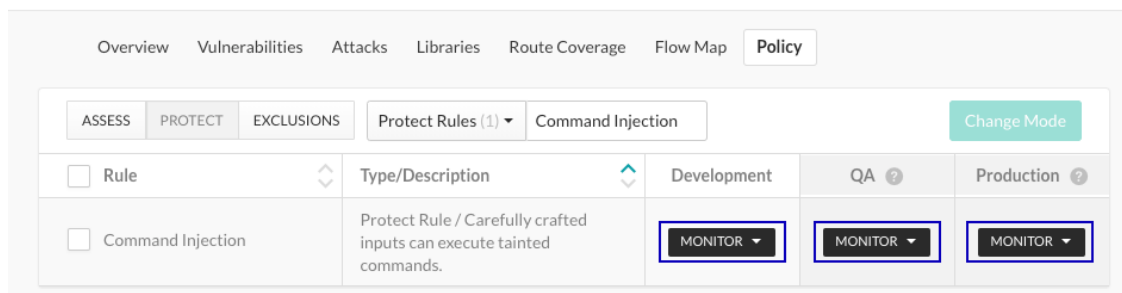
This option is available if you are using Java, .NET Framework, .NET Core, Python, Node.js, Go, or Ruby agents.

- a. In the Attack Events list, at the end of an attack events row, select the dropdown (▾).
- b. Select **Add Exclusion**.
- c. Specify a name for the exclusion.
- d. Select the exclusion type and enter the details for that type.
- e. Select the rules for which the exclusion applies.

To see a list of rules, click the **Applicable rules** box.



- f. (Optional) Select the checkbox to suppress all events that match the exclusion.
 - g. Click **Add**.
6. Create a virtual patch (attack events):
- [Virtual patches \(page 1138\)](#) are short-term, custom defense rules that defend against specific, newly-discovered vulnerabilities in your code.
- a. In the Attack Events grid, at the end of an attack events row, select the arrow.
 - b. Select **Create Virtual Patch**.
 - c. In [Add Virtual Patch \(page 1138\)](#), enter the details for the virtual patch.
 - d. Click **Save**.
7. Specify modes for Protect rules (attack events):
- [Protect rules \(page 1130\)](#) let you monitor or block specific kinds of cyber-attacks in application environments.
- a. In the Attack Events list, select an event.
This action displays the details for the attack event.
 - b. Select the settings icon (⚙️) next to the event name.
 - c. As needed, change the modes for each rule by selecting **Change Mode** or the **current mode** or a specific environment.



- d. Select the appropriate modes for each environment.

8. Save attack data:

Contrast keeps attack event data for thirty days before removing it. You have several options for saving your data:

- Output the data to [syslog](#). (page 918)
- Set up a [generic webhook](#) (page 1076).
A generic webhook can receive notifications on any URL that receives POST messages.
- Export the data to a CSV or XML file.
At the end of an attack row, select the arrow at the end of the row and select **Export attack (CSV)** or **Export Attack (XML)**.



Add tags to attacks ☺



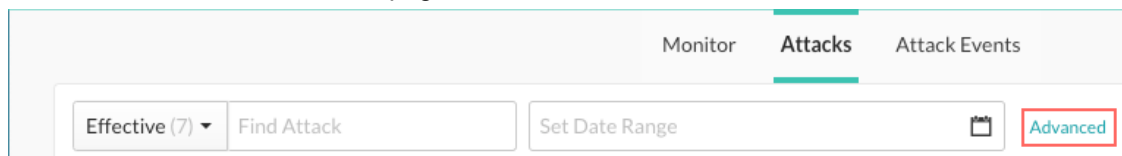
NOTE

This procedure is for on-premises customers.

To make it easier to find specific attacks or attack events, use tags.

Steps

1. Add tags to attacks or attack events:
 - a. Select **Attacks** in the header.
 - b. Select **Attacks** or **Attack events**.
 - c. Select the check box next to one or more attacks or attack events and select the tag icon (🔖).
 - d. In the Tag attacks or Tag attack event window, select an existing tag or create one.
 - e. Select **Save**.
2. Use tags to find attacks or attack events:
 - a. On the Attacks or Attack events page, select **Advanced** next to the search boxes.



- b. Expand the Tags section.

Advanced ✕ Clear ✕

☐ Show suppressed

☐ Show bots blocked

☐ Show denylisted

Tags (1 selected) ▲

☒ webgoat (1)

Status ▼

Rules ▼

Applications ▼

Application Importance ▼

Attackers ▼

Servers ▼

Environments ▼

Severity ▼

- c. Select the check box next to one or more tags.
The display changes to show the attacks or attack events that have the selected tags.

Use Contrast

The way you interact with Contrast depends on your particular situation, the tools and integrations you use, your roles and permissions, and whether you are accessing Contrast through the web interface, command line tools or the [REST API](#).



NOTE

All commands used in this guide should be run in a command shell with administrative privileges from the directory in which Contrast was installed.

The majority of Contrast users will likely be assigned an Editor role. (You can see what [permission level you have](#) (page 599) under [user settings](#) (page 596).)

With Editor permissions, you can [instrument an application](#) (page 52), start viewing results in Contrast, and interact with the basic components of Contrast (all visible in the header of the web interface):



NOTE

If your organization has the new role-based access control turned on, your administrator can tell you what [permissions](#) (page 1278) you have and what you can access.

- [Projects](#) (page 600)
View projects and findings for open source software after you run the CLI on a manifest, or you connect a GitHub, Bitbucket, or GitLab account to your Contrast organization.
- [Applications](#) (page 602)
View a searchable list of an organization's applications. License, merge, tag, archive and restore applications.
- [Scans](#) (page 632)
Run static scans and view results for Java binaries and multi-language source code using SAST technology.
- [Servers](#) (page 911)
View a searchable list of an organization's servers. Designate server environment, enable Assess and Protect, settings, tagging and deleting.
- [Libraries](#) (page 922)
View a searchable list of libraries being used by all the applications in an organization. Use tags and view statistics for known vulnerabilities present in libraries and high-risk libraries.
- [Vulnerabilities](#) (page 1021)
View a searchable list of vulnerabilities discovered. You can view this list for each application in an organization. Mark status, merge, share, tag, and export vulnerabilities. View details of any vulnerability for more information and guidance for fixing it.
- [Attacks](#) (page 584)

View a searchable list of attacks that are occurring or have occurred on all the applications in an organization. View attacks at the highest level or delve into the individual attack events.

You can also use other features and tools to enhance your Contrast experience:

- [Reports \(page 1043\)](#)
Collect data and export as a CSV or PDF to share it outside of Contrast.
- [Integrations \(page 1051\)](#)
Use Contrast in conjunction with other tools like bugtrackers, build tools, application servers, Security Incident Event Management (SIEM), notifications and chat.
- [Contrast CLI \(page 994\)](#)
Perform software composition analysis (SCA) on your application to show you the dependencies between open-source libraries.

Although most of the configuration for these features requires [system \(page 1185\)](#), [organization \(page 1153\)](#) or [RulesAdmin \(page 1118\)](#) permissions, an Editor can:

- [Instrument an application \(page 52\)](#)
- [Send notifications \(page 599\)](#)
- [Customize scoring \(page 1177\)](#)

Supported languages for Contrast

This table provides a view of the languages the different Contrast components support.

Table key:

● Supported
● Under consideration
-- Not applicable for this technology

Language	Contrast Scan	Contrast Runtime SCA	Contrast Assess	Contrast Protect	Contrast Application Detection and Response (ADR)	Contrast Serverless	Application Vulnerability Monitoring (AVM)
Java	●	●	●	●	● Includes observability	● (AWS and Azure) Includes SCA library data, SAST scans, and IAM policies	●
Kotlin	●	●	●	●	●	--	●
Supported by the Java agent							
Scala	●	●	●	●	●	--	●
Supported by the Java agent							

Language	Contrast Scan	Contrast Runtime SCA	Contrast Assess	Contrast Protect	Contrast Application Detection and Response (ADR)	Contrast Serverless	Application Vulnerability Monitoring (AVM)
.NET Framework	●	●	● (Azure functions) (page 327)	●	●	● (Azure functions) (page 327) Includes SCA library data and IAM policies	●
.NET Core	●	●	● (Azure functions) (page 327)	●	●	● (Azure functions) (page 327) Includes SCA library data and IAM policies	●
C# Supported by the .NET Core and .NET Framework agents	●	●	●	●	●	● Includes SCA library data and IAM policies	●
VB.NET Supported by the .NET Core and .NET Framework agents	●	●	●	●	●	--	●
Node.js	●	●	●	●	●	● Includes SCA library data and IAM policies	●
Client-side JavaScript	●	--	--	--	--	--	--
Ruby (version 3.2.X or earlier) Visit Maintenance status for Ruby .	●	●	●	●	●	--	--
Python	●	●	●	●	●	● Includes SCA library data and IAM policies	●
Go	●	●	●	●	●	--	●
PHP	●	●	●	●	●	--	●

Additional technologies for Contrast Scan

Contrast Scan supports these additional languages and technologies:

SAP ABAP	JavaScript/TypeScript	RPG IV
ActionScript	JCL	Swift
ASP.NET	JSP	Transact-SQL
C and C++	NATURAL	TypeScript
COBOL	Objective-C	Visual Basic
Hana SQL Script	Oracle Forms	XML
HTML	PS-SQL	
Informix (SQL and 4GL)	PowerScript	

Contrast dashboard

Typically, when you log in to Contrast, you see the Contrast dashboard. The dashboard provides a high-level view of the state of your application risks to vulnerabilities and attack threats. It also provides trend information for vulnerability remediation and attacks.

Dashboard details

- **Summary:** The summary at the top of the dashboard provides these details
 - **Application Score:** This section shows the Custom Code score, the Library score, and the Overall score. The [Application scoring guide \(page 1269\)](#) describes how Contrast calculates these scores.
 - **Score trend:** This value shows whether your application score is improving.
 - **Attack trend:** This value shows whether your level of risk due to attacks is improving.
 - **This week's status for applications:** This value shows the number of deployed applications and the number of applications at risk for vulnerabilities for the current week.
 - **Library status:** This value shows the number of libraries discovered and the number of libraries at risk for vulnerabilities.
 - **Server status:** This value shows the number of known servers and their status.
- **Average time to remediate:** This section shows the current Average Time to Remediate (ATTR) vulnerabilities with a critical, high, and medium severity. It also shows the ATTR for critical, high, and medium vulnerabilities over time in a graph.
On-premises customers: Use the filter to select a specific time frame to refine the displayed details. When you use this filter, the graph and counts update based on the following criteria:
 - Vulnerabilities in a closed state.
 - Vulnerabilities with a last seen date within the date range.
 - The average time to remediation (ATTR) from the set of vulnerabilities. Contrast calculates this value by looking at the difference between the first seen and closed dates on the selected vulnerabilities, and averaging them.
- **Status and trends for vulnerabilities:** This section shows the number and types of vulnerabilities that Contrast discovered and are not remediated. It also shows the remediate trend.
For the vulnerability status, use the filter to refine the view by severity, status, or type.
For the vulnerability trend, use the filter to set a time frame for the trend details.
- **Attack status and trends:** This section shows the number of attacks and the trend for attacks that Contrast has seen.
For the attack status, use the filter to refine the view by status or type.
For the attack trend, use the filter to select a specific time frame for the trend details.

Manage user settings

To manage user settings in Contrast, open the **user menu** (your name in the top right corner of the Contrast web interface), and select **User settings**. There you can:

- [Change your password \(page 597\)](#)

- [Set up two-step authentication \(page 597\)](#)
- [Edit your profile \(page 598\)](#)
- [Find API keys \(page 598\)](#)
- [Manage notifications \(page 599\)](#)
- [View your permissions \(page 599\)](#)

Log in to Contrast

To log in to Contrast for the first time, you must accept an email invitation generated by your administrator. Select the link in the email to log in to Contrast.

If your organization is using [single sign-on \(SSO\) \(page 1169\)](#), select the checkbox on the login page to disable the password input field. You are only required to enter your email address. Once your email is verified you can log in with your full SSO credentials.

If you are using [two-step authentication \(page 597\)](#), your login process occurs *after* successful SSO authentication.

Change your password

To change your account password, complete the following steps.

1. Log in to Contrast.
2. In the user menu, select **User settings > Change password**.
3. In the form fields, enter your current password and new password. Retype your new password in the next field to confirm it.



NOTE

Your new password must adhere to the password policy [set by your administrator \(page 1168\)](#). Contrast notifies you of their requirements as you begin typing.

4. Check the box to agree to the [Terms and Conditions](#).
5. Select **Save**.



NOTE

Customers using single sign-on (SSO) to [log in \(page 597\)](#) don't have the option to change their password.

Set up multi-factor authentication

If your administrator has enabled multi-factor authentication at an [organization \(page 1168\)](#) or [system \(page 1232\)](#) level, you can add an extra layer of protection beyond your username and password. To set it up:

1. In the user menu, select **User settings > Multi-factor verification**.
2. Use the toggle to enable multi-factor authentication.
If an administrator has disabled multi-factor authentication at an organization level, you cannot turn it on.
3. Use the radio buttons to select how you want to receive verification codes:
 - **Email:** You can receive authentication codes in the email you associated with Contrast. To set this up, you will receive an email with a verification code to enter in the configuration page

- **Google Authenticator:** You will need to download the app to your mobile device, and you can receive authentication codes there. To set this up, scan the QR code provided in Contrast, and follow the instructions to validate your device.
4. Before completing multi-factor authentication setup, you can download a set of backup codes in the form of a `.txt` file, which allows you to login if you encounter an error or get locked out of your account. You must download and save these codes in a secure location.
 5. If you want to change the way you receive verification codes, you can go back and reconfigure notification settings. Once you change your selection, Contrast automatically issues a new set of backup codes. It is not necessary to save your changes.

**TIP**

If you run into issues using either method, you can use the backup codes provided, select **Can't sign in?** or use the **Reset Device** link in Google Authenticator.

Manage your profile

Add or update your name, time, date and language preferences to customize your experience using Contrast.

1. In the user menu, select **User settings > Profile**.
2. Under **General information**, you can enter your basic information, such as your name or time zone.
3. Click on the thumbnail to upload a new profile image.
4. Here you can also [view your organization and personal keys \(page 598\)](#) under **Your keys**.
5. Select **Save**.

View your API keys

Use API keys to establish communication between agents or custom scripts and Contrast. The agent and the Contrast API use the keys for these purposes:

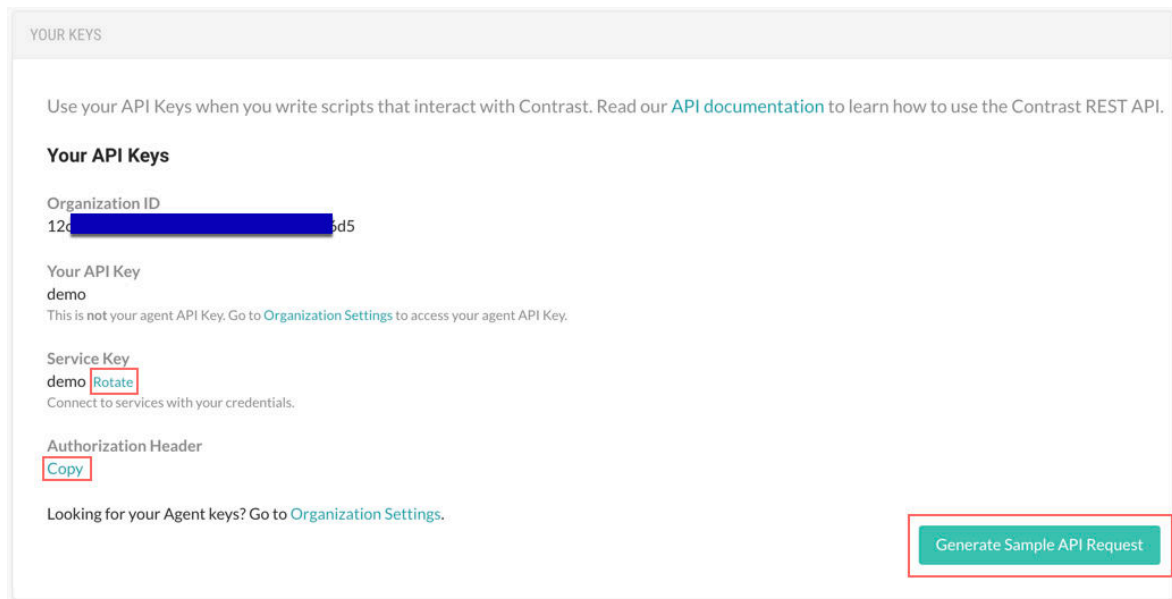
- To identify which organization is being accessed.
- To identify you as a valid user.

Steps

1. Go to **User menu > User settings > Profile**.
Under **Your keys** you can see the **API keys**, which include the API key and these keys:
 - **Organization ID:** Identifies the organization being accessed.
 - **Service key:** Uses your credentials to connect to Contrast services.
 - **Authorization header:** Identifies you as a valid user.This key contains this value: `base64(username:service_key)`

**NOTE**

The Organization ID and API key are shared across all users. Only the service key is unique to each individual user.



2. To copy the authorization header, select **Copy**.
3. To rotate the service key, click **Rotate**.
Rotating the service key affects any integrations using that key. To reconnect, update your credentials in your integrations.
4. To generate a sample API request and copy it to clipboard, select **Generate sample API request**.

Manage user notifications

To change your notification settings:

1. In the **user menu**, select **User settings > Notifications**.
2. Click in the **Subscriptions** field to choose the application(s) for which you want to receive notifications. The default selection is "All Applications".
3. Use the toggles in the **In Contrast** and **Email** columns to enable or disable the following subscriptions.
 - **Active attack:** There is an active attack on an application with Protect enabled.
 - **New vulnerability:** Contrast has detected a new vulnerability. Click in the field to receive notifications for specific severity levels or "Library"; the default selection is "All".
 - **Vulnerable library:** Get notifications when a new vulnerable library is detected or a new CVE is detected in a library.
 - **Server messages:** Get server messages around reliability issues.
 - **Server offline:** A server can no longer be reached.
 - **New comment:** A team member commented on a finding.
 - **New asset:** A new asset (application or server) to which you have [access \(page 599\)](#) has been added. Click in the field to set this notification for "Application" or "Server"; the default selection is "All".
 - **Nearing expiration:** An application license is about to expire.
 - **Policy violations:** A compliance, library or remediation policy is in violation. Select the box if you want to aggregate policy violation emails into a digest.
 - **Email:** A daily summary of Contrast activities.

View your permissions

The **Permissions** page provides a detailed view of the assigned permissions for both the organization and the applications to which you have access. To see your permissions:

1. Go to the **User menu > User settings > Permissions**.

2. See your organization listed at the top of the page along with your organization role.
The **Application permissions** grid shows your role for each application within the organization.
3. Click the help icon next to each role for details on the data access and actions made available by each level.

See also

- [Application roles \(page 1264\)](#)
- [Organization roles \(page 1267\)](#)

Projects

After the CLI is run on a [manifest](#), or a GitHub, Bitbucket, or GitLab account has been connected to your Contrast organization, you can view the associated projects and findings in Contrast. This provides earlier and broader visibility of vulnerabilities in open-source software where instrumentation is impossible or too late in the software development life cycle (SDLC).

To view CLI results on this page, you must have the Contrast CLI 2.0 [installed via npm \(page 995\)](#).

To view repository scan results on this grid, you must [connect your accounts \(page 602\)](#) to Contrast.



NOTE

Connections to Bitbucket and GitLab are available by request only. [Contact Support](#) to enable the connections.

See also

- [View projects \(page 600\)](#)
- Supported [languages and package managers \(page 995\)](#) for Contrast CLI
- [Libraries and SCA \(page 935\)](#)

View projects

There are multiple ways to view information about Projects and repositories connected to Contrast from GitHub, Bitbucket or GitLab.



NOTE

- Depending on your permissions in the organization, you may or may not be able to perform actions on this page. To be able to delete a repository you will need [admin permissions \(page 1264\)](#) enabled.
- Connections to Bitbucket and GitLab are available by request only. [Contact Support](#) to enable the connections.

Projects All (710)

Sort by
Name

Connections

Name	Last Activity Date	Vulnerable Libraries	
> contrast-security-app-test/team-xyz-test-java-not-protected-repo-202306... 1 project	07/07/2023 01:56 PM Branch updated	None identified	
> contrast-security-app-test/team-xyz-test-java-not-protected-repo-202306... 1 project	07/07/2023 01:56 PM Branch updated	None identified	
> contrast-security-app-test/team-xyz-test-java-not-protected-repo-202306... 1 project	07/07/2023 01:56 PM Branch updated	8 (44)	
> contrast-security-app-test/team-xyz-test-java-not-protected-repo-202306... 1 project	07/07/2023 01:56 PM Branch updated	None identified	
> contrast-security-app-test/team-xyz-test-java-not-protected-repo-202306... 1 project	07/07/2023 01:57 PM Branch updated	8 (44)	
> contrast-security-app-test/team-xyz-test-java-not-protected-repo-202306... 1 project	07/07/2023 01:57 PM Branch updated	8 (44)	
> contrast-security-app-test/team-xyz-test-java-protected-repo-1 1 project	07/07/2023 01:57 PM Branch updated	8 (44)	
> contrast-security-app-test/team-xyz-test-java-protected-repo-10 1 project	07/07/2023 01:43 PM Branch updated	8 (44)	
> contrast-security-app-test/team-xyz-test-java-protected-repo-2 1 project	07/07/2023 01:38 PM Branch updated	8 (44)	
> contrast-security-app-test/team-xyz-test-java-protected-repo-20230628... 1 project	07/07/2023 01:57 PM Branch updated	None identified	

Page 26 of 71

Results per page 10 25 50

- Select **Projects** in the header.
- Select a project name from the list and, if applicable, expand the row to view more information about the connected projects. Each row contains the latest activity and the total number of vulnerable libraries along with the total number of critically vulnerable libraries.
- The Projects view shows:
 - **Name:** This is the name of the project containing either the manifest stored locally for the CLI or the GitHub, Bitbucket, or GitLab account and repository name. The type of project is identified by the analysis performed.
 - Projects analyzed by Contrast CLI are identified with this icon .
 - Projects analyzed by the Contrast Security GitHub App, Bitbucket, or GitLab are identified with this icon .
 - **Last activity date:** The latest activity (the reason for the trigger) and a timestamp of the latest activity.
 - **Connections:** [Connect \(page 602\)](#) Contrast to your GitHub, Bitbucket, or GitLab account to see results.
 - **Vulnerable libraries:** This shows the number of libraries in the project with an identified vulnerability (CVE). The number of libraries with at least one critical severity CVE is coded red. If applicable, expand a row to view the connected projects. Hover over the thermometer section to see the number of CVEs by severity. Click the thermometer to open the details panel. If vulnerabilities exist, they are displayed in a list and are color-coded by severity.
 - If a message of *Analysis not complete. Try again or contact Support.* appears, it means that the analysis is in progress or there is a failure. Try connecting the repository again. If that fails [contact Support](#) for help.
 - If a message of *None identified* appears, it means that no vulnerable libraries have been found.
 - **Actions:** This is where you can view the repository, [disconnect \(page 602\)](#) the repository connections, or [export \(page 602\)](#) the CLI project data to a CSV file.
- Use the *sort by* dropdown at the top-right to sort the list by the last activity date or the repository name.


Export project details

You can export project details in a CSV file or in a generated Software Bill of Materials (SBOM) file.

The exported CSV file contains data fields like *Library Name*, *Language*, *Version*, *Release Date*, *CVE Count*, and *Licenses* for each project.

Steps

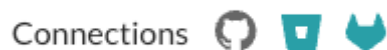
To export project details:

1. Select **Projects** in the header.
2. Locate the row for the project to be exported.
3. Select the **Export** icon  at the end of the row.
4. Select the format under the *Download As* options. If you select the SBOM option you will need to specify the standard.

The file will be downloaded to your desktop.

Account connections

With the connection options at the top of the Projects grid, you can easily connect Contrast to your GitHub, Bitbucket, or GitLab repositories.



Once you connect to either platform, scan results will appear on the Projects grid in Contrast. You can manage individual connections row-by-row on the Projects page.



NOTE

Connections to Bitbucket and GitLab are available by request only. [Contact Support](#) to enable the connections.

To learn more about the repo connections and how they work, see [Libraries and SCA \(page 935\)](#).

Applications

After an [application has been instrumented \(page 52\)](#), you can [view your applications \(page 603\)](#) in Contrast and explore these features:

- [Route coverage \(page 619\)](#)
- [Session metadata \(page 616\)](#)
- [Flow maps \(page 631\)](#)
- [Application scoring \(page 1269\)](#)



NOTE

An Organization or Application Administrator will need to [license your application \(page 1161\)](#) for these features to work.

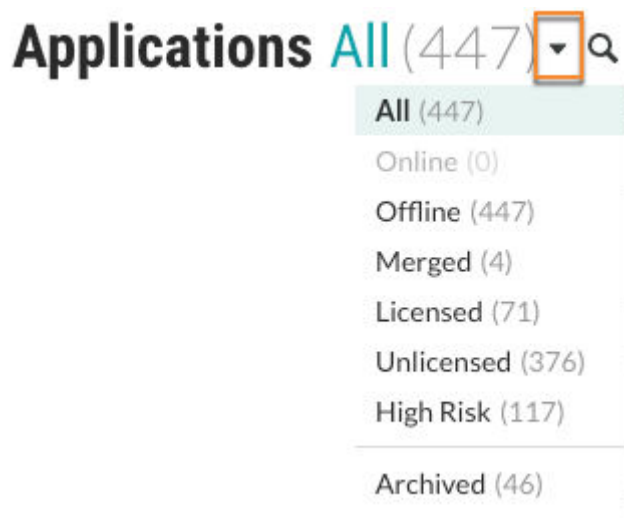
View applications

There are multiple ways to view application information.

Steps

1. Select **Applications** in the header to view a list of all applications found in your organization.
2. Select an application name from the list to view the application's **Overview** tab.
3. To filter the list based on application status, select the small triangle at the very top of the applications list.

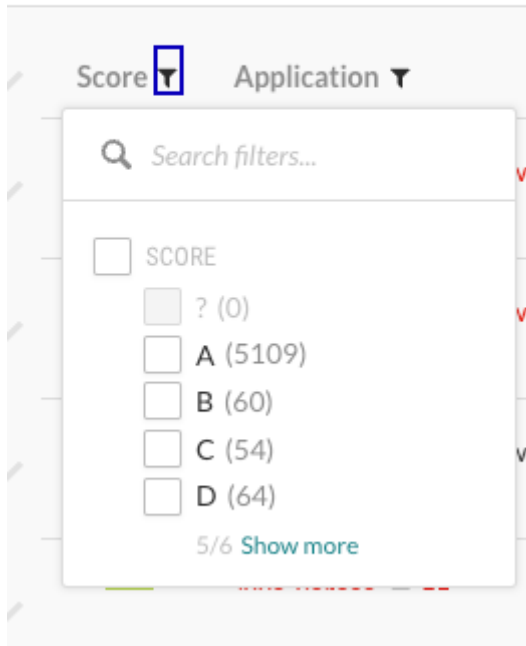
Alternatively, to search for specific application by name, select the **magnifying glass** icon (🔍) to search for them.



The filters include:

- **All:** Applications that you added to Contrast, excluding archived applications.
 - **Online:** Only applications whose agents have contacted Contrast within the last 5 minutes. Excludes archived applications.
 - **Offline:** Only applications whose agents have had no contact with Contrast for more than 5 minutes. Excludes archived applications.
 - **Merged:** Only applications that are part of a merged application (the primary application and its components). Excludes archived applications.
 - **Licensed:** Only applications that have a Contrast license applied to them. Excludes archived applications.
 - **Unlicensed:** Only applications that have no Contrast license applied to them. Excludes archived applications.
 - **High risk:** Only applications that have a high or critical vulnerability with a status of: Reported, Suspicious, or Confirmed. Excludes archived applications.
 - **Policy violation:** This filter is available when the organization has enabled compliance policies. Only applications that are in violation of a compliance policy. Excludes archived applications.
 - **Archived:** Only applications that are visible for historical purposes. The agent for an archived application no longer reports vulnerabilities to Contrast.
4. To filter by application score, select the **filter** icon (▼) next to the Score column header and select one or more scores.
To remove filters, select **Clear** next to the column header.

Applications All (53)



5. To filter the list of applications, select the **filter** icon (▼) next to the Application column header. The filters include:
- **Application metadata (if configured):** Application metadata that you associated with applications.
 - **Application tags (if created):** Tags you assigned to applications.
 - **Languages:** The language that an application uses.
 - **Servers:** The servers where applications are running.
 - **Open vulnerability severity:** The severity for open vulnerabilities.
 - **Technologies:** Technologies that applications use. For example, JSON or jQuery. To remove filters, select **Clear** next to the column header.
 - **Environments:** The environments associated with applications: Development, QA, and Production.
 - **Application importance:** The importance level you set in the application settings.

Applications All (5301) 🔍

Score ▼	Application ▼
A	<input type="checkbox"/> Search filters...
A	<input type="checkbox"/> BUSINESSUNIT
A	<input type="checkbox"/> Death Star Security (1)
A	<input type="checkbox"/> TESTNUMERICFIELD
A	<input type="checkbox"/> 421 (1)
A	<input type="checkbox"/> TESTPOINTOFCONTACTFIELD
A	<input type="checkbox"/> TK-421 (1)
A	<input type="checkbox"/> APPLICATION TAGS
A	<input type="checkbox"/> (3)
A	<input type="checkbox"/> 28052021 (2)
A	<input type="checkbox"/> abc (4)
A	<input type="checkbox"/> abc123 (5)
A	<input type="checkbox"/> app-other-tag (6)
A	5/1208 Show more
A	<input type="checkbox"/> LANGUAGES
A	<input type="checkbox"/> Java (4809)
A	<input type="checkbox"/> .NET Framework (206)
A	<input type="checkbox"/> .NET Core (22)
A	<input type="checkbox"/> Node (115)
A	<input type="checkbox"/> Ruby (39)
A	5/9 Show more
A	<input type="checkbox"/> SERVERS
A	<input type="checkbox"/> 0035f200-f1b6-11ec-8a53... (9)
A	<input type="checkbox"/> 01c8dd20-f1ad-11ec-b32a... (9)
A	<input type="checkbox"/> 023b1f90-f1b0-11ec-b415... (10)
A	<input type="checkbox"/> 023b1f90-f1b0-11ec-b415... (10)
A	<input type="checkbox"/> 028da1d0-f1af-11ec-afa4-... (10)
A	5/1171 Show more
A	<input type="checkbox"/> OPEN VULNERABILITY SEVERITY

Application details

The Overview tab for an application shows details about the application configuration and activity.

Summary

The summary at the top of the tab shows this information:

- **Scores:** The letter grade is based on the number and seriousness of the vulnerabilities found in the application. Use the [application scoring guide \(page 1269\)](#) for guidance.
- **Libraries:** The number of open-source libraries that Contrast SCA identified and the number of vulnerable libraries.
- **Routes exercised:** The first value is the number of routes exercised in the application. The second value is the number of observed routes.
To view [route coverage \(page 621\)](#) details, select the value for the exercised routes.
- **Servers:** The number of servers associated with the selected application.
- **Routing frameworks:** The supported frameworks that Contrast detected during route discovery.
During an agent session, Contrast [checks its compatibility \(page 621\)](#) with the frameworks it detects in your code.

Environment details

For each environment, you see these details, if servers are defined for the application:

- **Protect and Assess settings:** The setting bars indicate the number of servers where Assess or Protect is turned on or off.
To manage the settings, select a section of the bar. Doing so opens a filtered [view of the servers \(page 912\)](#) where you can manage the Assess and Protect settings.
- **Servers:** The number of servers associated with the application.
- **Vulnerabilities:** The number of vulnerabilities. Use filters to change the view by severity, status, or type.
Hover over a section of the vulnerabilities bar to view additional details.
- **Vulnerability trend:** This view shows the trends for vulnerabilities that Contrast identifies and the remediation of these vulnerabilities, either by manual verification or auto-verification policies.

Additional details

The Applications page contains these additional tabs with application details:

- [Vulnerabilities \(page 1024\)](#)
- [Libraries \(page 922\)](#)
- [Route coverage \(page 619\)](#)
- [Flow map \(page 631\)](#)
- [Policies for Assess rules \(page 1119\)](#)
- [Policies for Protect rules \(page 1130\)](#)

Contrast Security Observability

Contrast Security Observability models an application's security architecture and behavior at runtime.

Use this information to better understand the underlying behavior of your applications for threat modeling, pen test support, and contextual information around vulnerabilities and attacks.



NOTE

Currently, this feature is supported for Java applications only.

Benefits

Once you add Contrast with security observability turned on for your applications, services, and APIs, you can quickly view in-depth details about their behavior and actions, such as:

- Security defenses by resource, such as authentication (authn) and authorization (authz)
- Potentially dangerous calls made during each of those resource calls (for example, SQL or system commands)
- Backend connections for each resource

You can use this information for:

- Threat modeling data flow diagrams
The observed data provides a ground-truth view into the behavior and architecture of your application that most impacts its security posture. You get:
 - Lightweight instrumentation of applications that offers continuous monitoring for high-performance examination of real application behavior.
 - Runtime visibility into application architecture, for example: Identification of all services, APIs, and files that the application accesses.
- Penetration testing support
The observed data about your endpoints can help you effectively target your tests. You get:
 - Contextual behavior based on application utilization
 - Run time visibility in to application architecture
- Vulnerability prioritization
The observed data provides information about a vulnerability's context to help you determine its impact and exploitability. You get:
 - Identification of services, APIs, files, and databases that the application accesses.
- Attack context
The observed data provides context when a service or API is under attack. This data can help you determine its impact and exploitability.

Observe mode configuration

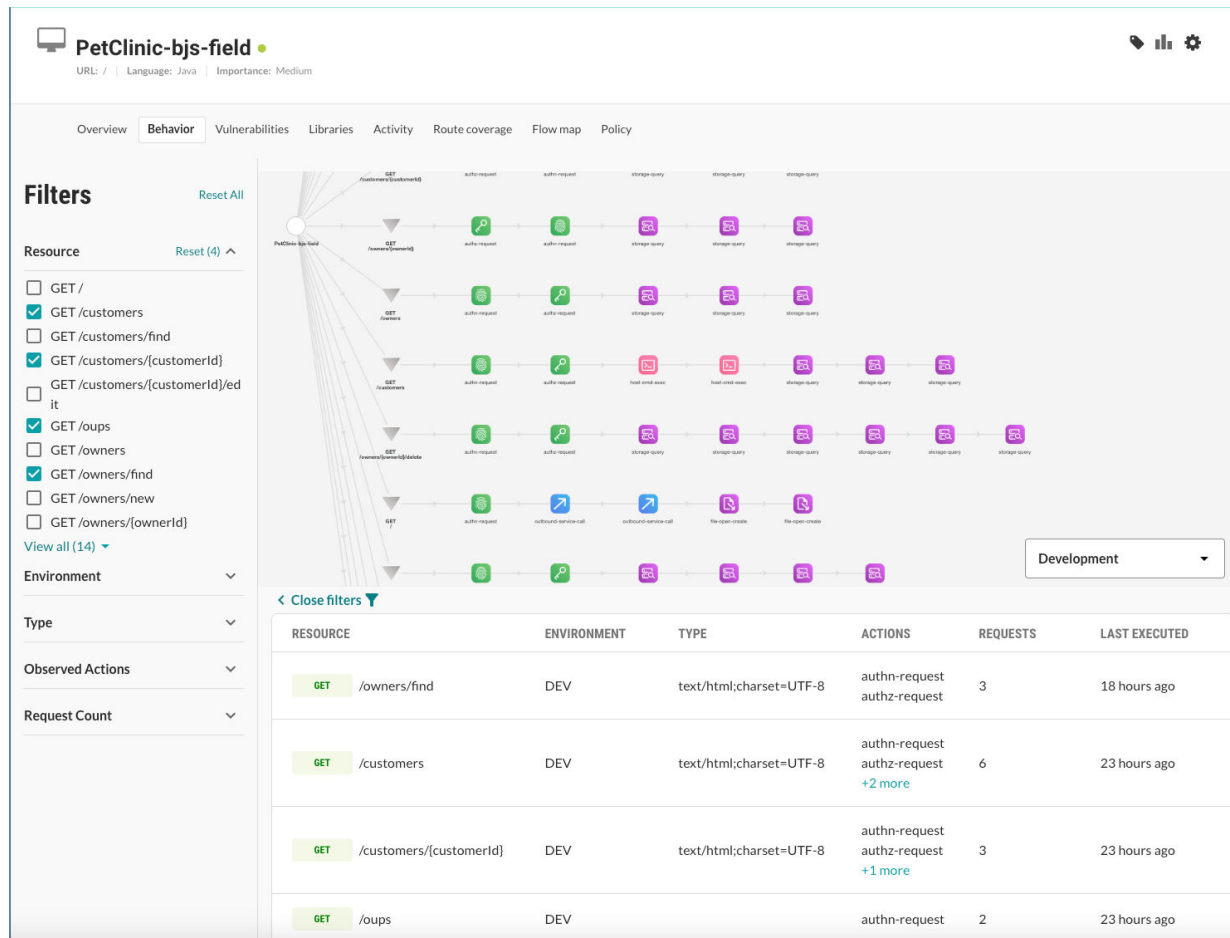
The observe mode setting in your agent's configuration file turns on the ability to view security observability details:

```
observe:  
  enable: true
```

Once observe mode is turned on and you exercise your application, [view the behavior insights \(page 608\)](#) from the Applications page in the Contrast web interface.

Runtime behavior insights

The runtime behavior insights that security observability provides includes these key details:



- **Application model:** An graphical representation of the behavior and connections to other components that Contrast detects in an application.
- **Resource:** Identifies an application's entry points, such as routes used, message queues, web requests, and so forth.
- **Type:** The response type that a resource uses. For example: `application/json` or `text/html`.
- **Actions:** Security-relevant behavior that Contrast observes in the application, including:
 - Database connections, including the instance name of the database
 - File system access, including the name of the file or directory accessed
 - Outbound service or API calls
 - Authentication and authorization detection
 - System commands
 - Potentially dangerous functions
 - Security controls: Shows which of your service or API routes have custom security controls applied.
- **Environment:** The environment where an action occurs: Development, QA, or Production.
- **Requests:** The number of requests made for the resource.
- **Last executed:** The last time a resource was exercised in the application.

See also

[View runtime behavior \(page 608\)](#)

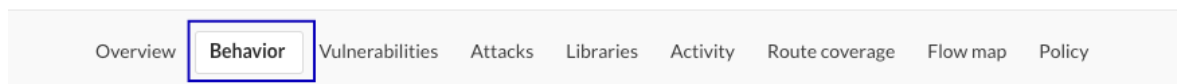
View insights for runtime behavior

Before you begin

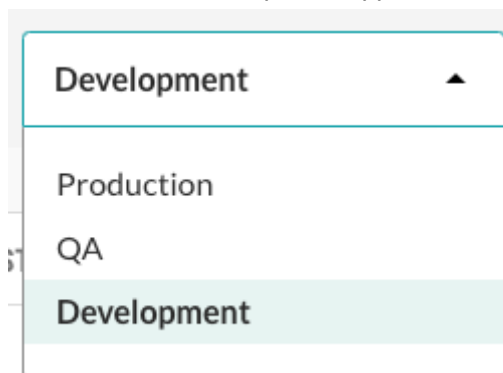
- Verify that the [observe mode setting \(page 607\)](#) is turned on in the agent's configuration file.

Steps

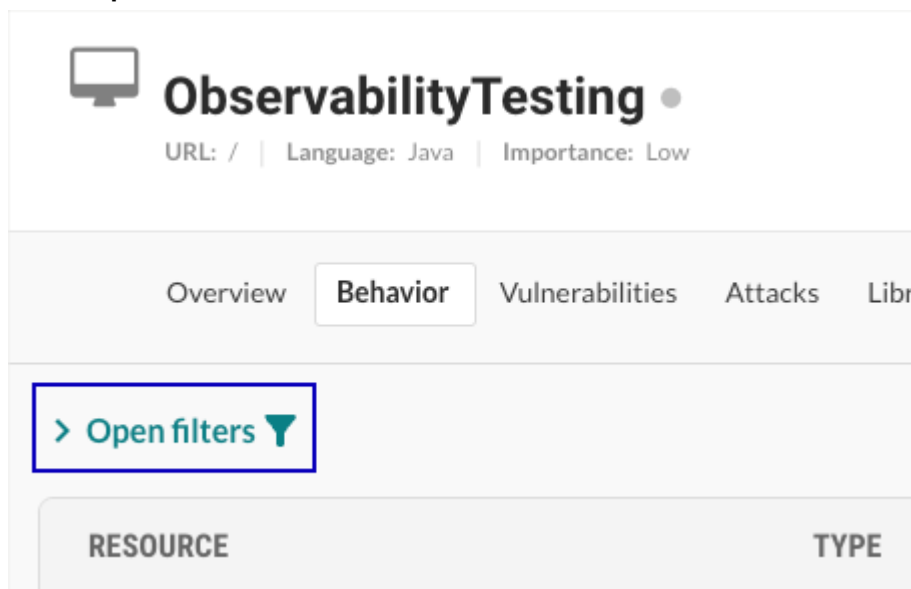
1. Select **Applications** from the header
2. Select an application.
3. Select **Behavior**.



4. To view details for a specific application environment, select it in the dropdown above the list.

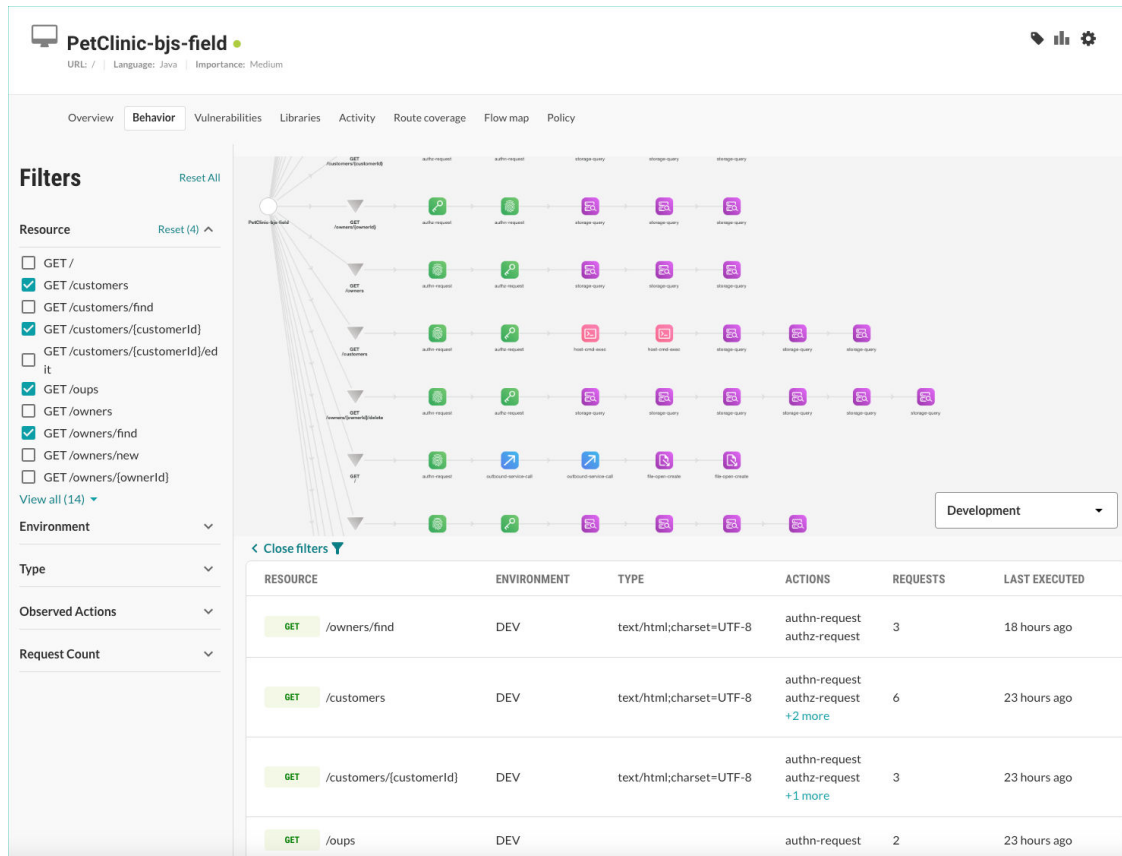


5. To refine the view with filters:
 - a. Select **Open filters**



- b. Select all the filters you want to use. The filters are:
 - **Resource:** Select one or more resources.
 - **Environment:** Select any or all of these environments: Development, Production, and QA
 - **Type:** Select any or all data exchange types.
 - **Observed actions:** Select the actions you want to view from these categories. For example, Authn-Request, File-Open-Create, and Storage-Query.

- **Request count:** Select a count grouping, from 0-1000 up to 100000+.



6. To view additional details, select a resource in the list.

This selection opens a panel that shows a model of the resource behavior and connections to other components that Contrast detects. It also shows details for the actions, including database queries and outgoing API calls


When you select an element in the model, the list highlights the actions associated with that element.

The screenshot shows the Contrast Security interface for an application named 'PetClinic-bjs-field'. The 'Applications' tab is active, displaying a flow map of the application's behavior. A sidebar on the left contains filters for resources, environment, type, and request count. The main area shows a detailed view of a specific endpoint: 'GET /customers/{customerId}'. This view includes a flow map of the request, environment details (DEV), and a list of actions performed by the application, such as 'AUTHN-REQUEST' and 'STORAGE-QUERY'.

Edit application settings

You may need to access or edit settings for your application:

- **Application names:** Each application in an organization must have a unique name. To change the name, select Applications in the header, then click on the application's name in the grid to go to its **Overview** page. Click on the name at the top of the page to update the text. Alternatively, select the **Settings** icon in the top right of the Overview page and update the name in the **Application defaults** window. SuperAdmins can also edit application names by selecting **SuperAdmin** in the user menu, then **Applications** in the header, then clicking on the name in the grid.
- **Application ID:** The application ID is the last URI segment in the URL of your browser. To locate an application's ID, select an application from the grid. The segment after `applications/` is the application ID.
- **Application importance:** This value appears in the application's metadata and may also be used in your organization's integrations settings. To set an application's importance level select an application name to view its **Overview** page and select the **Settings** icon. In the **Application defaults** window, use the **Importance** field to select a level from the dropdown.

1. Select your application from the **Applications** tab. Your selected application overview displays.
2. Select the settings  icon in your application overview. The **Application Settings** popup displays.
3. Edit any of the fields as required.
4. Click **Save** to save your updated application settings.

Field descriptions

- **Application names:** Each application in an organization must have a unique name. To change the name, select Applications in the header, then click on the application's name in the grid to go to its **Overview** page. Click on the name at the top of the page to update the text.
Alternatively, select the **Settings** icon in the top right of the Overview page and update the name in the **Application defaults** window.
SuperAdmins can also edit application names by selecting **SuperAdmin** in the user menu, then **Applications** in the header, then clicking on the name in the grid.
- **Application Code:** The identification code or number that's internal to your organization and unique to the application or microservice. Use this field if you want to integrate these applications' unique identifiers into your usage of Contrast. This code can be configured upon application startup, in the application properties section of an agent configuration.
- **Override URL:** This is the url used to replicate a vulnerability.
- **Importance:** This value appears in the application's metadata and may also be used in your organization's integrations settings. To set an application's importance level select an application name to view its **Overview** page and select the **Settings** icon. In the **Application defaults** window, use the **Importance** field to select a level from the dropdown.


Add tags to applications

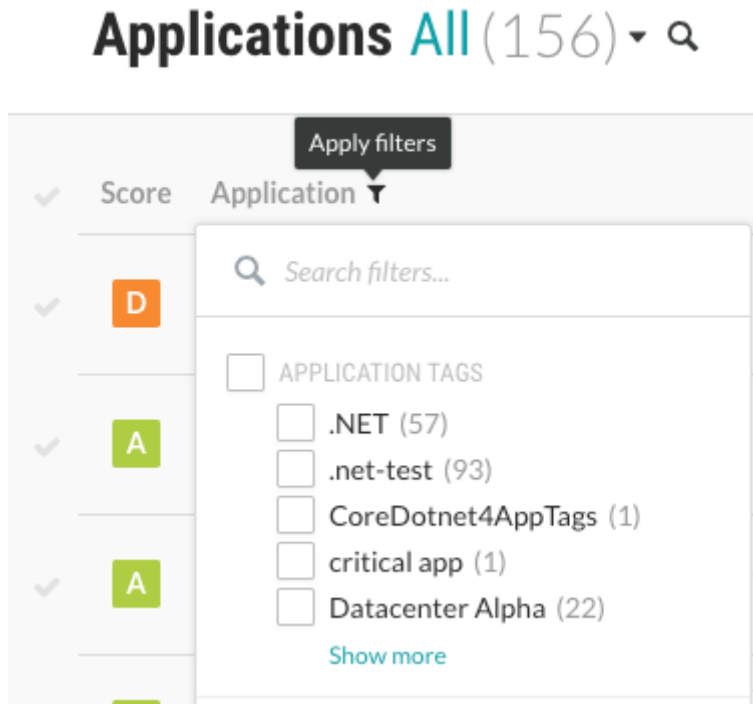
Use application tags to better organize applications and improve search functionality.

Steps

1. Select **Applications** in the header.
2. Add tags:
 - a. To add tags for a single application, hover over the end of the application's row and select the **Tag** icon (🏷️).
Alternatively, go to the Overview page for the application and select the **Tag** icon (🏷️) at the top of the list.
 - b. To add tags for multiple applications, select the check mark next to each application and select the **Tag** icon from the action menu at the bottom of the list.



3. In the Tag application window, start typing to view a list of existing tags. Select the tags you want to use or enter a new tag.
After adding tags, you can see them next to the application name in the Overview tab for the application.
To remove a tag, click **x** in the tag name.

4. To use tags for filtering, select the Filter icon (▼) next to the Applications column and select Application tags.



Merge and unmerge applications

Merging two or more applications creates a single application called a primary application. Merging applications is a common operation for administrators responsible for bringing applications online.

The main purpose of merging is to present a single application view for the purposes of scoring, discovered vulnerabilities, and remediation. Merged applications can be made up of modules, which show up individually in the application list. Merging also allows you to logically organize all of an application's modules into one entity in Contrast.

Data access after merging

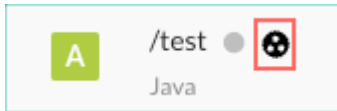
Access to data for merged applications depends on the type of Contrast access control you are using.

- **Legacy access control** (on-premises customers and customers who are not using the new role-based access control):
 - When you merge applications, an application module inherits the application group of the primary application.
 - To access data for the application module, users must be a member of the application group that includes the primary application.
- **New role-based access control** (customers with role-based access control turned on):
 - When you merge applications, access to application data is unaffected. Administrators can change user access to application data after you merge an application.
 - If a user has a role that includes an application module as a resource, they continue to have access to that application's data. If you remove the application module from a resource group associated with the role, users no longer have access to the data.
 - If a user has a role that doesn't include the application module as a resource, they have no access to the application's data. To provide access to the data, add the application module to a resource group associated with the role.

Steps

1. Select **Applications** in the header, and use the check marks to select the applications to merge.

2. Select the **Merge** icon (🔗) from the menu at the bottom of the list.
3. In the Merge applications window, use the dropdown to choose one of the merged applications to represent the primary application.
4. Once your applications are merged, you see the primary application icon (⊕) beside the name of the primary application.



5. To see the application modules in the merged application as well as details about exercised routes, select the primary application icon in the application's row .

Application Modules (2) ×

/test is a LICENSED application that represents 2 modules.

<input type="checkbox"/>	Grade	Application	Routes Exercised
<input type="checkbox"/>	A	/test ⊕ ● Java	7 / 7
<input type="checkbox"/>	A	app-01a057e4-5f7a-478c-af13-3b1763fe2ce7 ● Java	4 / 4

Cancel Unmerge Selected

6. To unmerge either all or specific application modules from the primary application, select the primary application icon (⊕) in the application's row or Overview page. In the Applications modules window, select any number of the modules, and select **Unmerge selected**.

Archive and unarchive applications

If an application should no longer collect vulnerabilities, but you want to keep it in your organization for historical purposes, the best solution is to archive the application.

Archiving an application maintains the integrity of past application data, such as vulnerabilities and libraries, but the agent no longer reports vulnerabilities to Contrast.

Archived applications also improve your overall portfolio score, as they don't count against the total score.

An administrator can restore an archived application. After you unarchive an application, it is visible in the default Applications list. All vulnerabilities and issues immediately impact the its score.

Before you begin

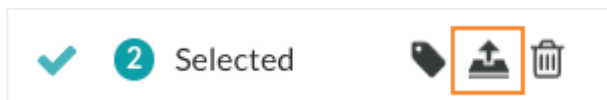
- Archiving an application doesn't free up a license in Contrast. To return a license back to the pool of available licenses is to archive and fully [delete \(page 616\)](#) the application.

Steps

1. Select **Applications** in the header.
2. Find the application you want to archive.
3. Archive the application:
 - a. To archive a single application, hover over the end of the application's row and select the **Archive** icon (📁).
 - b. To archive multiple applications, select the checkmark next to each one and select the **Archive** icon (📁) from the action menu at the bottom of the list.



- c. In the displayed window, select **Archive** to confirm your choice.
4. To unarchive an application::
 - a. Select the small triangle (▾) next to the Applications header at the top of the list.
 - b. Select **Archived**.
 - c. To unarchive a single application, hover over the end of the application's row and select the **Unarchive** icon (📁).
 - d. To unarchive multiple applications, select the checkmark next to each one and select the **Unarchive** icon in the action menu at the bottom of the list.



- e. In the displayed window, select **Unarchive** to confirm your choice.

Reset an application

Resetting an application purges all the data associated with it, but doesn't remove the application. Resetting applications is useful when you want to clear all history and findings associated with a specific application.

It is common to reset an application before [deleting \(page 616\)](#) it to make sure that all associated vulnerabilities, URLs and components are cleared properly.

Reset behavior

- **General behavior:**

When you reset an application, it loses its historical vulnerability data, library data, and route coverage data. It keeps its license association and entry in Contrast, as well as its server associations.

- **Application or server is online:**

If you reset an application that is online or its server is online, the count for vulnerabilities, libraries, and routes becomes 0. If you browse some of the application endpoints, data for vulnerabilities, libraries, and routes repopulate. Discovered routes remain lost, because route discovery occurs during application startup.

- **Application or server is offline:**

If you reset the application when the application or its server is offline, all the data populates, as expected.

Before you begin

You can reset only one application at a time.

**CAUTION**

If you reset an application, there is no way to restore the purged data.

Steps

1. Select **Applications** in the header.
2. Hover over the end of the row and select the **Reset** icon (↺),
Alternatively, in the application's Overview tab, select the **Settings** icon (⚙️) and select **Reset application**.
3. In the Reset Application window, select **Reset**.

Delete applications

When you delete an application, Contrast permanently removes all of its associated findings, such as vulnerabilities and libraries.

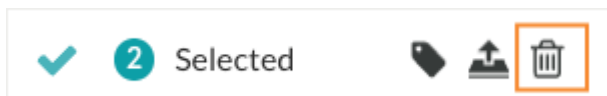
Licenses applied to applications permanently count towards the number of maximum allowable applications. Deleting a licensed application has no effect on the number of licenses you are allowed to apply to applications

Before you begin

- **Optional:** You can [reset the application \(page 615\)](#) before you archive and delete it.
Resetting an application purges all of the data associated with it, but doesn't remove the application. Users often reset applications to clear all history and findings associated with a specific application.

Steps

1. Select **Applications** in the header.
2. Find the application that you want to delete.
3. Archive the application:
 - a. To archive a single application, hover over the end of the application's row and select the **Archive** icon (📁).
 - b. To archive multiple applications, select the check mark next to each one and select the **Archive** icon (📁) in the action menu at the bottom of the list.
 - c. In the displayed window, select **Archive**.
4. Select the **Dropdown** icon (⌵) next to the Applications header at the top of the page and select **Archived**.
5. Delete the application:
 - a. To delete a single application, hover over the end of the row of the application you want to delete and select the **Delete** (🗑️) icon.
 - b. To delete multiple applications, select the check mark next to each one and select the **Delete** icon (🗑️) in the action menu at the bottom of the list.



- c. In the displayed window, select **Delete**.

Use session metadata filters

Use session metadata to filter vulnerabilities and route information for a specific branch, build, committer, or repository. When you [add the necessary configuration settings \(page 618\)](#) to your agent

configuration file, the agent reports this information along with the rest of your standard vulnerability data to Contrast.

A session is the combination of metadata values that you set in the agent configuration file. Depending on the defined values, each agent run could be part of a single session or every agent run could have its own session. If you are integrating Contrast into a CI/CD pipeline, ensure that you send at least one session metadata value that is unique each time you deploy a new version of the application. For example, configure the agent to send the Commit Hash or Build Number metadata because these values are likely to change for each application deployment.

If you don't select a specific session metadata filter, the Session column in the Vulnerability list displays up to 10 of the values specified in the agent configuration file. This limit ensures that the Contrast web interface can display vulnerability and sink group data correctly.

The session metadata filters that you apply affect the Vulnerabilities and Route coverage lists.

Auto-generated session metadata

When a Contrast agent starts, it automatically creates a unique fingerprint that is equal to that unique build of an agent and populates session metadata values. The key used for this feature is called `artifactHash`.

You can use this metadata in the same way as the ones you create manually.

The agent versions that support this feature are:

- Java 6.11.1 and later
- .NET Core 4.3.9 or later
- .NET Framework 51.1.9 and later
- Python 9.7.0 and later
- Node.js 5.24.0 and later

Steps

1. Select **Applications** in the header.
2. Select an application in the list.
3. Select either the **Vulnerabilities** tab or the **Route coverage** tab.
 - In the Vulnerabilities tab, select the Session metadata icon (🔍) at the top of the list.
 - In the Route Coverage tab, in the Session routes section of the Route coverage summary, select **Apply filter**.
4. Select the session: **Most recent session** or **Custom session** and the filter.

In the Vulnerabilities list in the Vulnerabilities tab, the Session column and the View by filter are hidden and the details of the most recent session display above the session metadata icon.

 - a. In **System property**, select one of the displayed properties.
 - b. In **Value**, start typing to find values for the selected system property.

Session metadata filtering

Specify the session metadata of Adam Webgoat Route Session Metadata Test to display historical views.

☐ Most recent session
☒ Custom session


System property
Committer

Value
8/25/2021, 4:50:10 PM - Build Number: 8.0.1, Repository: webgoat, Committer: Jane, Branch Name : feature/some-n...

Clear filter
Apply filter

5. Select **Apply filter** to save the selected filter.

6. To clear a session metadata filter, use one of these methods:


- In the Vulnerabilities tab, In the Vulnerabilities list, select the Session metadata icon (). and select **Clear filter**.

Session metadata filtering

A session metadata filter was applied to Adam Webgoat Route Session Metadata Test. To specify a different session, select Clear filter.

Current session: 8/25/2021, 4:50:10 PM
Build Number: 8.0.1
Repository: webgoat
Committer: Jane
Branch Name : feature/some-new-thing

Clear filter
Apply filter

Alternatively, select **Clear** next to the displayed metadata above the Session metadata icon ().

Current Session: 8/25/2021, 4:50:10 PM
Clear

↓ SORT BY LAST ACTIVITY

In the Vulnerabilities list, the Session column displays after you clear the session metadata filter.

- In the Route Coverage tab, in the Session metadata section of the Route coverage summary, select **Clear filter**.

Configure session metadata

To send session metadata for your application to Contrast, add session metadata key-value pairs to your agent configuration file.

The agent reports the following build properties. You may include all or some of these properties. When you do, the metadata will be available to you as additional information for each vulnerability reported or as a way to filter them.

Supply these settings as system properties, environment settings or properties in a YAML configuration file.

Name	Setting
Commit Hash	commitHash
Committer	committer
Branch Name	branchName
Git Tag	gitTag
Repository	repository
Test Run	testRun
Version	version
Build Number	buildNumber

The metadata string format must be an RFC 2253 compliant string of comma delimited key=value pairs. Do not include these characters:

- A space or hash (#) characters at the beginning of the string.
- A space at the end of the string
- Special characters: comma (,) , plus sign (+), double quotes ("), backslash (\), left angle bracket (<), right angle bracket (>), or semicolon (;)

Here are some examples of how you might configure session metadata in the following instances:

- **Java system properties:** Include an additional entry in the line where you add your `javaagent` flag. In this case, you will set the property `contrast.application.session_metadata` to a set of key-value pairs (which conform to RFC 2253) that identify your test run.

```
-Dcontrast.application.session_metadata="branchName=feature/some-new-thing,committer=Jane,repository=Contrast-Java"
```

- **.NET Framework using *app.config* or *web.config*:** You can add an entry to your configuration to specify this property.

```
<?xml version="1.0"?>
<configuration>
  <connectionStrings />
  <appSettings>
    <add
key="contrast.application.session_metadata" value="branchName=feature/
some-new-thing,committer=Jane,repository=Contrast-DotNet" />
  </appSettings>
</configuration>
```

- **YAML configuration:** You can add an additional entry to your `contrast_security.yaml` file.

```
application:
  session_metadata: branchName=feature/some-new-thing,committer=Jane,repository=Contrast-Ruby
```

- **Continuous integration (CI) build scripts:** You can set values using environment variables.

```
-Dcontrast.application.session_metadata="branchName=feature/some-new-thing,committer=Jane,repository=Contrast-Java,buildNumber=$BUILD_NUMBER"
```

```
-Dcontrast.application.session_metadata="branchName=$GIT_BRANCH,committer=$GIT_COMMITTER_NAME,commitHash=$GIT_COMMIT_HASH,repository=$GIT_URL,buildNumber=$BUILD_NUMBER"
```

Route coverage

For Assess users, route coverage associates vulnerabilities with the originating web request.

With route coverage, you can see detailed information on the components of your application, such as which routes were exercised and which ones were not. This information can help you decide where to focus testing and remediation.

Web request example

Web requests are the primary interface of web applications. A request may be handled by one function with many subsequent functions coordinating interactions with other services, databases, or files.

During the request handling process, Contrast monitors data flows across the application to identify vulnerabilities. A single web request may be vulnerable to multiple types of attacks. Contrast associates these vulnerabilities with the original request.

This example shows a web request:

```
GET /users?active=true
Host: YourDomain.com
Accept: application/json
```

This example shows how a function might handle the web request:

```
@Controller
public class UserController {
    @GetMapping("/users")
    public String users(@RequestParam(name="active", required=false,
    defaultValue=true) Bool active) {
        ...
    }
}
```

How route coverage works

An application route is a combination of three parts:

- An HTTP verb (for example: `GET`).
- The resource path (for example: `/users`).
- The method signature of the controller (for example: `UserController.users(Bool active)`).
For cases where using a method signature isn't appropriate, Contrast uses route templates, such as `route.jsp`, `route.xhtml`, or `/route/id/{id}`.

When a Contrast agent starts, it instruments functions in the application so that the agent can assess web requests for vulnerabilities while the application is running. If a function implements a framework to handle web requests, Contrast can identify the route before a request is handled. In Contrast, the status for these routes is **Discovered**.

When your application is handling a request, Contrast tracks the activity as an **Exercised** route.

Frameworks and technologies

Contrast supports route discovery for these frameworks:

- **Java:** all [currently supported technologies \(page 120\)](#)
- **.NET Framework:** all [currently supported frameworks \(page 212\)](#)
- **.NET Core:** all [currently supported frameworks \(page 271\)](#)
- **Node.js:** all [currently supported frameworks \(page 332\)](#)
- **Python:** all [currently supported frameworks \(page 410\)](#)
- **Ruby:** all [currently supported frameworks \(page 471\)](#)
- **Go:** all [currently supported frameworks \(page 530\)](#)

If the framework you are using is unsupported, [contact Support](#). For unsupported frameworks, Contrast will attempt to infer the routes based on observed requests, but you will not see any routes discovered within Contrast.

Exclusion of built-in routes and applications

Contrast route coverage excludes built-in routes in select web frameworks and applications. For example:

- The Jersey framework for Java applications includes a built-in route for serving a WADL file. Contrast does not include this route in its route coverage. Other web frameworks have similar built-in routes.
- The Contrast Java agent does not report routes from built-in applications such as the Tomcat Manager Application.

Compatibility checking for routing frameworks

When you instrument an application with a Contrast agent, Contrast looks for routing frameworks that it supports. Supported routing frameworks help Contrast to report detailed data about vulnerabilities and route coverage.

If your application has unsupported frameworks, you may be able to see vulnerability data but other features like route coverage or dedupe may not be available for your application.



NOTE

These agent versions support this feature:

- Java 6.3.0 and later
- .NET Core 4.2.14 and later
- .NET Framework 51.0.32 and later
- Node.js 5.9.0 and later
- Python 8.2.0 and later

How compatibility checking works

During an agent session, the Contrast agent discovers routes and checks for supported routing frameworks. Where possible, Contrast displays details about which frameworks it finds.

- If Contrast detects supported frameworks during route discovery, it displays a list of these frameworks.
- If the selected application is a merged application, Contrast displays routing framework details for the primary application only.
- If Contrast doesn't support the routing frameworks that your application uses, it displays a message that it can't detect routing frameworks.

If you are using older agents, Contrast might not be able to detect the routing frameworks. Consider updating your agent if you want to be able to view this information.

View routing framework details

1. Select **Applications** in the header.
2. Select an application.
3. In the Overview tab, in the Summary section, view the routing framework details on the right side.

View route details

[Route coverage \(page 619\)](#) helps you understand how vulnerabilities map to your application's attack surface.

If you remove routes from the Route coverage list (steps 8 and 9) and they still exist when the server restarts or you exercise the application, Contrast includes them in the list again. To permanently exclude routes, select the **Exclude** icon (🗑️) at the end of the route's row.

Steps

1. Select **Applications** in the header.
2. Select the name of an application.
The Overview tab shows the number of routes exercised compared to the number of total routes in your application.
3. In the Overview tab, select the number of routes exercised or select the **Route coverage** tab.

4. In the Route coverage tab, the Route coverage summary shows this information:

Session	Repo	Commit
6/15/2023, 9:16:36 ...	TS	commit-4-703669c0...
Developer		
Dev-4-703669c0-0b...		

- **Exercised routes:** This section shows these details:
 - The percentage of discovered routes that you exercised.
 - The number of exercised routes.
- **Session metadata:** This section shows these details, based on the applied session metadata filters:



NOTE

If you haven't applied session metadata filters, no values display. To see values for session metadata, select **Apply filter** or **Edit filter** (to change the current filters) and [specify the filters \(page 616\)](#) you want to use.

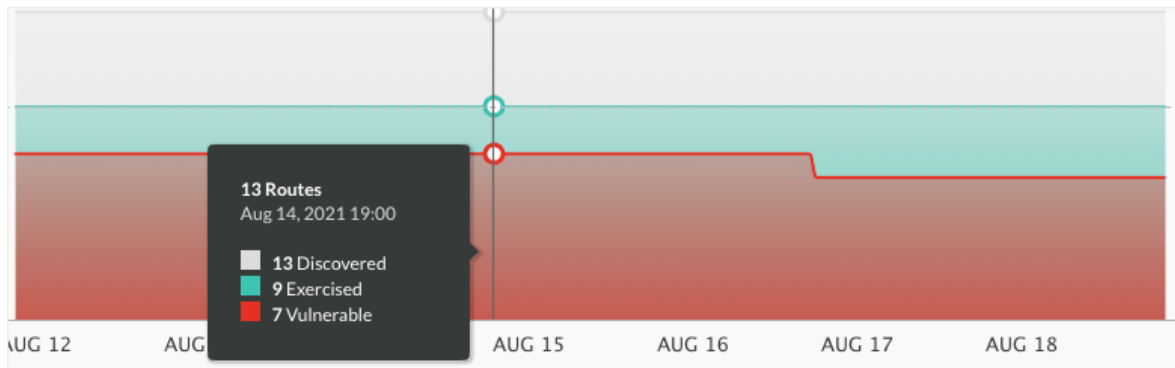
- Percentage of routes exercised that match the applied filters.
- The number of exercised routes that match the applied filters.
- The selected session.
- The repository for the application
- Build number
- Branch name
- Committer

5. In the Route Coverage tab, if you don't apply session metadata filters, the route coverage chart displays.



NOTE

If you previously selected metadata filters, this chart does not display. To see the chart, select **Clear the filters**.



The chart displays details about routes based on their status:

- Discovered by Contrast (but never exercised with the agent)
- Exercised with the Contrast agent
- Exercised and found to contain vulnerabilities

6. In the Route coverage list, view additional details about each route.

- **Route:** A route that Contrast identified or is tracking.
- **Server:** The servers where the application is running.

By default, the Server column shows up to three servers. To view a complete list of servers (if more than three are in use), select **Show all**.

Route	Environment	Server	Vulnerabilities	Application	First Seen	Last Activity	Status
✓ hello.ContactController.contactForm(org.springframework.ui.Model)	Development	CONTRAST-SVR1 CONTRAST-SVR2 CONTRAST-SVR3 Show all (5)	0	8885-test-app	2 months ago	1 minute ago	Exercised



NOTE

When you delete a server, Contrast removes it from the list instead of displaying it as greyed out.

- **Vulnerabilities:** The number of vulnerabilities associated with the route.
- **First seen:** The first time Contrast observed the route.
- **Last activity:** The activity time span for the route.
- **Status:** The route status.

7. Select an option to view details for each route that Contrast has identified in the application:
 - a. To view the URL for the route, select a route name.
 - b. To view vulnerability details for a specific route, select a section of the severity bar Vulnerability column. Each section indicates one or more severity levels: Critical, High, Medium, Low, and Notes.
 - c. To view routes based on the applications where Contrast observed them, select the Filter icon (▼) next to the Application column.
 - d. To view routes based on the time when Contrast first observed them, select the **Filter** icon (▼) next to the First seen column.

- e. To view routes based on an activity time span, select the **Filter** icon (▼) next to the Last Activity column.
Changing the time span also changes the time span for the route coverage chart.
To clear the filter selection, select **Clear** next to the column heading.
8. To remove a single route from the list:
 - a. Hover over the end of the row and click the **Remove** icon (🗑).
 - b. To confirm the removal of the route, select **Delete**.
9. To remove multiple routes from the list:
 - a. Select the check mark next to one or more routes or to select all routes, select the check mark next to **Route**.
 - b. In the batch action menu at the bottom of the page, select the **Remove** icon (🗑).
 - c. To confirm the removal of the route, select **Delete**.
10. To view and share route details outside of Contrast:
 - a. Select the check mark next to one or more routes or to select all routes, select the check mark next to **Route**.
 - b. In the batch action menu at the bottom of the page, select the **Export** icon (📄).
This action exports the details to a CSV file. The file downloads to your default download location.
The CSV file includes:
 - A list of the application's routes.
 - Details about the server on which they were found.
 - Details of when the routes were last exercised.
 - A list of vulnerabilities, the severity and status of each.

See also

- [Exclude routes \(page 625\)](#)
- [Include routes \(page 625\)](#)

Route exclusion and inclusion

In situations where you require a specific percentage of route coverage for security reasons, Contrast provides an option to exclude irrelevant or inaccessible routes from route coverage calculations.

You can choose to re-include any routes that you excluded previously.

Including or excluding routes requires an Admin role.

Effects of route exclusion

- Contrast collects vulnerability data for excluded routes, but excludes this data from application scoring calculations.
- Excluding a route excludes it from all environments where Contrast discovers it.
- If you defined session metadata to exercise and discover routes in an application, excluding a route also excludes this data.
- The audit log includes an entry for each route that you exclude.

Effects of route inclusion

- Contrast includes data for the included route in application scoring calculations.
- Including a route includes it in all environments where Contrast discovers it.
- The audit log includes an entry for each route that you include.

Exclude routes

Excluding routes (page 624) that are irrelevant or inaccessible helps to ensure that route coverage calculations for applications are accurate.

To ensure the route remains excluded, don't remove it after you mark it as excluded. If you exclude a route and then remove it, Contrast includes the route again if its detected when the application server starts or after you exercise the application.

Before you begin

- Identify the routes that you want to exclude.

Steps

- Select **Applications** from the header.
- Select an application name.
- Select the **Route Coverage** tab.
- Exclude one or more routes:
 - To exclude a single route, hover over the end of the row and select the **Exclude** icon (🗄️).
 - To exclude multiple routes, use the check marks in the left column to select routes. Then, select the **Exclude** icon from the batch action menu at the bottom of the page.



- Confirm the exclusion in the Exclude route window.
The status for the selected routes changes to **Excluded**.

Include routes

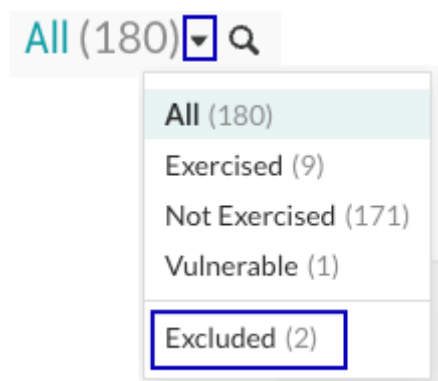
You can include routes (page 624) that you previously excluded. On the Route coverage page, excluded routes have a status of **Excluded**.


Before you begin

- Identify the excluded routes that you want to include.

Steps

- Select **Applications** in the header.
- Select an application name.
- Select the **Route coverage** tab.
- Display a list of excluded routes by selecting the **triangle** (▾) at the top of the list and select **Excluded**.



5. Include one or more excluded routes:
 - a. To include a single, excluded route, hover over the end of the row and select the **Re-include** icon ().
 - b. To select multiple excluded routes, use the check marks in the left column to select routes. Then, select the **Re-include** icon in the batch action menu at the bottom of the page.



- c. Confirm the inclusion in the Re-include route window.
The status returns to the status the route had before you excluded it.

Configure route expiration policy

Configuring a route expiration policy can ensure that your route coverage metric in Contrast is more accurate.

The policy affects route data that Contrast displays in the following ways:

- Routes expire only for active applications with discovered or exercised routes.
For route expiration purposes, Contrast views an application as active if it has had at least one route observed or exercised within the configured time period. For example, if you set the route expiration policy to 30 days, the application must have had at least one instance of route activity within the past 30 days for its routes to be eligible for expiration
- If Contrast does not see discovered or exercised routes in an active application after a specified number of days, the route is considered expired.
- After the specified expiration date occurs, Contrast deletes the expired route.
The route coverage calculation doesn't include the deleted routes.
- Before Contrast deletes expired routes, it sets the status of vulnerabilities associated with the route to **Remediated-Auto-Verified**.
- Routes do not expire if the application isn't active and has no discovered or exercised routes.

Before you begin

- The policy applies to all applications.
- Contrast applies the policy once per day.
Depending on the number of expired routes, it is possible that Contrast might not be able to delete all of them on the same day that they expire.

Steps

1. From the user menu, select **Organization settings**.
2. Select **Applications**.
3. In the Default section, under Route expiration policy, select the **Remove expired routes** checkbox and specify the number of days after which Contrast expires routes without activity.
The minimum value is one day. The maximum value is 365 days. The default value is 30 days.

Improve route coverage with production traffic

Before you begin

- Supported agent versions:
 - Java 6.6.0 and later
 - Node.js 5.20.0 and later
 - Python 9.4.0 and later

- .NET Core 4.3.0 and later
- .NET Framework 51.1.0 and later
- If you have any questions or concerns before using this feature, contact your Contrast representative for guidance.

Best practices

- Turn on IAST (Assess) in the application while it's in a pre-production environment before you deploy the application with IAST in the production environment. Doing so lets you check for compatibility and performance issues.
- Only use IAST in production environments for applications that use frameworks that Contrast Assess supports.
- Instrument singular instances of the application server.
If you use a load balancer that shifts traffic to different instances, it might be worth instrumenting one or more containers depending on the traffic routing rules.
[Limit agent deployments \(page 629\)](#) provides additional details for this situation.
- For performance impact and reduced overhead for IAST in production mode, reach out to your Contrast CSM. They have detailed data on average response time, average CPU usage, TPS, memory allocation, and more.
- For sensitive data masking, Contrast is not storing any sensitive data.
- Avoid changing the sampling settings in the agent configuration (YAML) file.
The settings in the agent configuration file override settings in the Contrast web interface. If you need to change sampling settings, it is best to use the web interface for this purpose.
In many cases, hosted customers don't have access to the sampling settings; Contrast manages them instead.

Turn on Assess in production environments

1. At the time of deployment, enable Assess in the agent configuration file.

```
Assess:
  enable: true
```

2. When Assess is enabled, deploy an application to Contrast and set **one** of these options:

- **System property:** `-Dcontrast.server.environment= PRODUCTION`
- **Environment variable:** `CONTRAST__SERVER__ENVIRONMENT= PRODUCTION`
- **Agent configuration file (YAML file):**

```
server:
  Environment: PRODUCTION
```

This option is case-sensitive.

The easiest method for deploying an agent is to use the Add New agent wizards in the Contrast web interface.

Alternative deployment

An alternative deployment option is to include the lower level sampling configurations for the agent. This case is applicable if you want to do testing in a lower environment and set the server environment appropriately. Follow these steps:

1. In the agent configuration file, leave the setting for Assess as `enable: false`. You can update this setting in the Contrast web interface if you need to turn off Assess analysis.
2. Update the agent configuration using environment variables or settings in the agent configuration file, as shown in these examples:

Environment variables:

```
CONTRAST__ASSESS__SAMPLING__ENABLE=true
CONTRAST__ASSESS__SAMPLING__BASELINE=20
CONTRAST__ASSESS__SAMPLING__REQUEST_FREQUENCY=2147483647
CONTRAST__ASSESS__SAMPLING__WINDOW_MS=3600000
CONTRAST__ASSESS__CACHE__VULNERABILITY__CACHE_PURGE_MS=3600000
CONTRAST__ASSESS__EVENT_DETAIL=minimal
CONTRAST__ASSESS__STACKTRACES=SINK
```

Agent configuration file (YAML file):

```
assess:
  event_detail: "minimal"
  stacktraces: "SINK"
  sampling:
    enable: true
    request_frequency: 2147483647
    baseline: 20
    window_ms: 3600000
  cache:
    vulnerability_cache_purge_ms: 3600000
```

Address performance issues

If you notice a negative performance impact after you enable Assess in production environments, change the [sampling rate frequency: \(page 916\)](#)

1. In the Contrast web interface, select **Servers** in the header.
2. Select the server where Assess is enabled in a production environment.
3. Select the Settings icon (⚙️).
4. Select **Enable sampling for higher performance**.
5. Change the **Analyzation frequency** to a lower value.



NOTE

Settings in the agent configuration file override settings in the Contrast web interface. When you turn on Assess for a production environment in the agent configuration file, the Contrast web interface shows the Assess setting as greyed out or turned off.

If you used the alternative deployment by setting the sampling settings, use this method to address performance issues.

1. In the Contrast web interface, select Servers in the header.
2. Select the server where Assess is enabled in a production environment.
3. Turn Assess to OFF
This setting takes effect only if Assess is not enabled in the agent configuration file.

Effect of using route coverage with production traffic

Using this feature does not result in any data loss. The effects of using this feature are:

- **Sampling:** Contrast analyzes routes only once in the Analyzation frequency window. Typically, this value is 24 hours.
- **Sensitive data:** Sensitive data is not returned to Contrast. There is no risk of storing real user data in Contrast. This behavior is typically preferred for security purposes.

- **Stacktrace and Contrast web interface differences (Java only and .NET only):** Java and .NET results show less depth in stacktrace data. In the Contrast web interface, all dataflow data is present, however red highlighting is not displayed.
- **Data tracing in events** Contrast doesn't highlight potentially exploitative payloads when it reports attack events. Contrast supports all other data tracing and analysis features.

Performance metrics

Use these performance metrics as a guide for understanding the potential impact of using Assess in your production environments.

For each agent, Contrast does performance testing in their SaaS environments with applications that may produce different results from applications in your environment. Your performance metrics may be different from Contrast metrics.

Language	Metrics
Java	The baseline impact for performance for the default settings increases the baseline of total request time by 2 ms against a baseline of no deployed agents.
.NET	The baseline impact increases the agent's total request time to 1.5 ms, compared to a baseline of .5 ms for Protect only, which already runs in production environments.
Node.js	The baseline impact to the test application increases the agent request time by an average of less than 10 ms compared to a baseline of no deployed agents.
Python	The baseline impact for performance for the default settings increases the baseline of total request time by 56 ms against a baseline of no agent.

Limit the number of agent deployments

When you deploy agents to monitor applications with IAST (Assess), you want consider their impact on performance. In cases where your web traffic is load balanced across a farm of servers, you may not need to install an agent on every server. If your load balancer distributes traffic evenly and routes all traffic through every server in the farm, you could achieve the same level of monitoring or vulnerability detection by deploying an agent to just one server. Similarly, you could have sample points at each back-end destination based on your load balancing topology.

This topic provides examples of how you can limit the deployment of agents to improve performance.

Reasons for limiting agent deployments

- **Performance optimization:** Deploying agents across all servers can add unnecessary overhead, especially if your applications are latency-sensitive or resource-constrained. By reducing the number of deployed agents, you minimize CPU and memory usage, which can improve response times and overall application performance.
- **Sufficient traffic coverage:** Since your load balancer evenly distributes traffic across all servers, an agent on one server would still monitor representative traffic. In many cases, this can provide enough data for security and monitoring purposes without the need to instrument every instance.

Deploy selected agents in Kubernetes (K8s)

In a Kubernetes environment, you can manage which pods in a deployment are instrumented with an agent by controlling the number of replicas that run with the agent. Use this strategy as a guide:

1. **Nodes taints:** Use node taints and tolerations to dedicate specific nodes for pods with agents. Label one node with a specific taint and schedule the agent-enabled pod to run on that node, leaving the others unaffected.
2. **Pod affinity:** Use pod affinity and anti-affinity to control where your agent-enabled pods run. You can create an affinity rule that only schedules the agent-enabled pod to a particular node or pod, ensuring it doesn't replicate across the entire farm.

3. **Deployment overrides:** Use a custom deployment manifest where you set up two deployments: one for the regular application pods and another for the agent-enabled pods. For the agent-enabled deployment, you can set replicas: on to ensure only one instance runs with the agent.

Example: Selective agent deployment in K8s

In this YAML example, only one replica of the application has the agent enabled.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: app-with-agent
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: my-app
        agent: enabled
    spec:
      containers:
        - name: app-container
          image: my-app-image
          env:
            - name: ENABLE_AGENT
              value: "true"
```

Deploy selected agents with Docker Swarm

Similar to what you can do with K8s, you can use service scaling and constraints in Docker Swarm, as described in this strategy:

1. **Service constraints:** Use service constraints to deploy the agent-enabled container to a specific node or set of nodes. For example, you can label a node and ensure that only one agent-enabled service runs on that node.

Bash example

This example deploys the agent-enabled instance to a node with the label `agent == true`, leaving other nodes unaffected.

```
docker service create \
  --replicas 1 \
  --constraint "node.labels.agent == true" \
  --env ENABLE_AGENT=true \
  my-app-image
```

2. **Task replication:** In Docker Swarm, you can use two services: one with the regular application and one with the agent. Scale the regular service across all nodes and the agent-enabled service to just one replica.

Bash example

This example shows how creates a setup where most of the containers run without the agent, but one instance includes the agent for monitoring purposes.

```
docker service create --replicas 5 my-app-image # Regular service
docker service create --replicas 1 --env ENABLE_AGENT=true my-app-image
# Agent-enabled service
```

Troubleshooting and testing recommendations

Deploying the agent to only one server in a web farm can introduce issues with traffic routing. Since your load balancer distributes traffic across all servers, you can't always guarantee that test traffic

will consistently go to the server with the installed agent. Verifying results can be more difficult, because some traffic may bypass the instrumented server. You may observe incomplete data or gaps in monitoring because not all traffic is routed to the correct server.

In such cases, you need to investigate your load balancer settings: Consider the following :

- **Session affinity (sticky sessions):** Ensure that your load balancer supports session affinity, which can route repeat requests from the same client to the same server. By enabling sticky sessions, you can direct all test traffic from a specific user or IP address to the server with the agent, improving consistency during testing.
- **Targeted testing:** You might need to use a load balancer feature that allows for weighted routing or manual targeting of specific servers for certain traffic. This feature would let you send your test traffic to the server with the agent, ensuring that the agent processes it.

Flow maps

The application flow map provides an interactive view of where data and resources are shared within your organization and beyond it.

Every time you exercise an application, Contrast uses data reported from your Contrast agent to create a detailed diagram of your application, the layers of technologies within it and the back-end systems to which it connects. As more applications are exercised within your organization and their back-end systems are identified by the agent, Contrast also identifies which applications are connected to the application you're currently viewing by shared back-end systems.

When you [view a flow map \(page 631\)](#), you can see the entire landscape of systems and resources that are associated with the application. By focusing on connections between individual systems and applications, you can also determine if users and connected applications in your organization have appropriate access to the current application and sensitive data potentially associated with it. Learn more about [understanding flow maps \(page 632\)](#).

The agent performs application matching through string credentials. Other instrumented applications that share common string credentials - for example, REST endpoints, database connection, or other unique host and port combinations - are displayed as connected applications.



NOTE

Users who [don't have access \(page 1164\)](#) to view details for a connected application, won't see that application in the flow map.

View flow maps

The application flow map provides an interactive view of where data and resources are shared within your organization and beyond it.

To view the flow map for an application:

1. Select **Applications** in the header and select the name of an application.
2. Click on the **Flow map** tab.
3. Here you will see three connected sections: **Application Architecture**, **Back-End Systems** and **Connected Applications**. Learn more about [understanding flow map data \(page 632\)](#).

Understand flow maps

The application flow map provides an interactive view of where data and resources are shared within your organization and beyond it.

When you [view a flow map \(page 631\)](#) for an application, the information is organized into three connected sections:

- **Application architecture:** This section breaks down the view, presentation and service layers of the application's front end. You can also see foundational information about the application, including the environments in which it's deployed, letter grade, vulnerability statuses and attack status. There are three layers to this section:
 - **View:** This column displays the layer of technologies that determine what a browser sees and processes.
 - **Presentation:** This column displays the layer of libraries that generates the application view.
 - **Service:** This column displays the layer comprised of the database, LDAP driver or back-end code performing the application logic.

Hover on an item in any of the lists to see how many instances of each type of library are used in the application, or click on the library to go to the library's page. If the agent reports any vulnerabilities, a warning icon appears beside the library in which they were found; hover over the icon for links to the vulnerabilities' **Overview** pages.

- **Back-end systems:** These columns display each of the systems to which your application is connected. Hover on the cylinder icon for databases, the globe icon for URLs, or the plug icon for LDAP databases to see more details on each system; click on an icon to highlight its connection to other applications. A solid line with lock indicates that the connection is encrypted; a dashed line shows that the connection is unencrypted or the state of encryption is unknown.
- **Connected applications:** This column lists each of the applications that are connected to the primary application by a back-end system. To see connected applications that meet specific criteria, click the funnel icon to select filters from the dropdown, such as environment, application language and custom tags. The menu also shows session metadata fields for the primary application (not the connected applications), if available. Select **See Flowmap** to go to the **Flow Map** tab for that application.



NOTE

If the agent isn't currently reporting data for the current application, the **Back-end systems** and **Connected applications** sections are left blank.



TIP

If the application is being accessed by another user while you're viewing the flow map, the **Browser** tab appears with a list of the browsers on which it's being accessed. Hover over the icons to see more details, such as the browser type and version.

Scans

Contrast Scan is a static application security testing (SAST) tool that lets you quickly scan code to identify vulnerabilities in early stages of development.

You can use these scan methods:

- **Hosted:** Use this scan method if you are able to upload code to the Contrast platform. To start a scan, use the Contrast web interface. Scan results are posted in the Contrast web interface.
- **CLI:** Use this scan method if you prefer to use CLI commands to upload code to the Contrast platform. Scan results are posted in the Contrast web interface or an integration such as GitHub or Jenkins.
- **Contrast Scan local engine:** Use this scan method for code on your local system. The Contrast platform receives the results but you don't upload local code. Scan results are posted in the Contrast web interface or in an integration such as GitHub or Jenkins.

Depending on the type of code you submit for scanning, Contrast Scan uses one of these scan engines:

- **Java binary:** Scans Java JAR or WAR files.
The Java binary scan supports only web applications (applications that handle HTTP traffic). This type of scan has a more narrow focus than a source code scan. It looks for data that comes from an untrusted source, such as user input and gets to a dangerous sink, like an SQL statement, without sanitization. The scan doesn't report on code that is not security relevant. This type of scan uses Scan policies (for example: the code contains dangerous potential sink calls or the calls or entry points allow untrusted data to enter the application) to find security-relevant code.
- **Source code** Scans artifacts for [most languages](#) ([page 657](#)).
This type of scan has a wider focus than a Java binary scan. It searches the code for potential vulnerabilities based on a rule set. The results are typically less accurate than a Java binary scan.

Scan feature comparison

This table lists the features that each scan method supports.

Features	Scan local engine	Contrast hosted platform	CLI
Scan types			
Multi-language source code scan	✓	✓	✓
Java binary	✓	✓	✓
Upload source code to Contrast platform	×	✓	✓
File size			
Max file size =1GB	×	✓	✓
Integrations			
SCM integration with GitHub action	✓	×	✓
Pipeline integration (for example, Jenkins)	✓	×	✓
Branch support	✓	×	×
Fail builds	✓	×	✓
Customizations			
Timeout settings	✓	×	×
Memory settings	✓	×	×
Resource group assignments	✓	✓	✓
File exclusions	✓	×	×

Scan tasks

In Contrast Scan, you can:

- [Run scans locally](#) ([page 672](#))
- [Create a scan project](#) ([page 665](#))
- [Archive a scan project](#) ([page 906](#))
- [Delete a scan project](#) ([page 667](#))
- [Monitor scans](#) ([page 669](#))
- [Analyze scan results](#) ([page 887](#))

- [Start a new scan \(page 668\)](#)
- [Cancel a scan \(page 669\)](#)
- [Change scan settings \(page 905\)](#)
- [Use Contrast Scan with GitHub repositories \(page 884\)](#)
- [Generate SAST Attestation report \(page 670\)](#)
- [Set notifications for failed scans \(page 1176\)](#)

See also

[Scan supported languages \(page 657\)](#)

Scan release notes

Scan source code engine

August 2024: Scan source code engine

Release date: August 29, 2024

These updates apply to the source code scan engine which you can use with the Scan local engine and a hosted deployment of Contrast.

New and improved:

- Made a significant number of improvements across languages to reduce false positives.
- Improved parsing of Natural source code to reduce false positives and improve scan performance.
- Improved parsing of COBOL source code to reduce false positives and improve scan performance.
- Updated the supported version of Kotlin to 1.6.0.
- Added support for scanning of Java 16 and 17 files.
- Improved support for Vue.JS.

Bug fixes:

- Fixed a bug that caused the source code scan engine to fail for all scans.

Scan engine and Scan local engine

Scan local engine 1.1.7

Release date: April 15, 2025

New and improved:

- Added new options to only fail a scan on a branch if it would introduce new vulnerabilities.

Bug fixes:

- Multiple fixes to improve Scan local engine behavior and performance.

Checksum:

- **MD5 checksum:** fa99a209ba3662a198df735fa4c795eb
- **SHA1 checksum:** 1c78f9570e20c18b01c4b609904f4bdf9cfe8eff
- **SHA256 checksum:** e4316485cba75bf032cfcd4537d1c9281bf8813bac03d84004e55b5bf415ec99

**NOTE****How to generate a checksum**

- **MD5:** Use the following command:

```
curl -L -H 'Accept: application/vnd.github.v3.raw' -s https://$CONTRAST_GITHUB_PAT@maven.pkg.github.com/Contrast-Security-Inc/sast-local-scan-runner/com.contrastsecurity.sast-local-scan-runner/X.X.XX/sast-local-scan-runner-X.X.XX.jar.md5 -o sastXX.md5
```

- **SHA:** Use the following command:

```
curl -L -H 'Accept: application/vnd.github.v3.raw' -s https://$CONTRAST_GITHUB_PAT@maven.pkg.github.com/Contrast-Security-Inc/sast-local-scan-runner/com.contrastsecurity.sast-local-scan-runner/X.X.XX/sast-local-scan-runner-X.X.X.jar.sha1 -o sastXX.sha
```

For both types of checksums, replace `x.x.xx` with the version of the engine you are downloading and validating with a checksum. For example, for the 1.1.0 version of the engine, replace `x.x.xx` with `1.1.0`, and for the output (`SastXX.sha` or `MD5`), a value to represent the current version, such as `10`.

Application signing verification

To verify that Contrast created and signed the Scan local engine that you downloaded, use this command:

```
jarsigner -verify -verbose -certs sast-local-scan-runner-0.0.XX.jar
```

Replace `xx` with the version of the Scan local engine that you want to verify.

Scan local engine 1.1.6

Release date: March 18, 2025

New and improved:

- Added these rules:
 - Improper Neutralization of Special Elements used in an SQLCommand (SQL injection) in [ASP.NET Core | Open-source web framework for .NET](#) applications.
 - Improper Control of Remote Code Execution for JavaScript.
- Updated Log4j libraries throughout codebase.
- Added the ability to identify and upload total lines of code and files scanned.

Bug fixes:

- Fixed a stack overflow error resulting from PHP scans under specific circumstances.
- Fixed an issue where the engine failed to parse some C# extension methods properly.

Checksum:

- **MD5 checksum:** fa99a209ba3662a198df735fa4c795eb
- **SHA1 checksum:** 1c78f9570e20c18b01c4b609904f4bdf9cfe8eff

- **SHA256 checksum:** e4316485cba75bf032cfcd4537d1c9281bf8813bac03d84004e55b5bf415ec99



NOTE

How to generate a checksum

- **MD5:** Use the following command:

```
curl -L -H 'Accept: application/vnd.github.v3.raw' -s https://$CONTRAST_GITHUB_PAT@maven.pkg.github.com/Contrast-Security-Inc/sast-local-scan-runner/com.contrastsecurity.sast-local-scan-runner/X.X.XX/sast-local-scan-runner-X.X.XX.jar.md5 -o sastXX.md5
```

- **SHA:** Use the following command:

```
curl -L -H 'Accept: application/vnd.github.v3.raw' -s https://$CONTRAST_GITHUB_PAT@maven.pkg.github.com/Contrast-Security-Inc/sast-local-scan-runner/com.contrastsecurity.sast-local-scan-runner/X.X.XX/sast-local-scan-runner-X.X.X.jar.sha1 -o sastXX.sha
```

For both types of checksums, replace `x.x.xx` with the version of the engine you are downloading and validating with a checksum. For example, for the 1.1.0 version of the engine, replace `x.x.xx` with `1.1.0`, and for the output (SastXX.sha or MD5), a value to represent the current version, such as 10.

Application signing verification

To verify that Contrast created and signed the Scan local engine that you downloaded, use this command:

```
jarsigner -verify -verbose -certs sast-local-scan-runner-0.0.XX.jar
```

Replace `xx` with the version of the Scan local engine that you want to verify.

Scan local engine 1.1.5

Release date: December 10, 2024

New and improved:

- Added support for Ruby source code scanning.

Checksum:

- **MD5 checksum:** fa99a209ba3662a198df735fa4c795eb
- **SHA1 checksum:** 1c78f9570e20c18b01c4b609904f4bdf9cfe8eff
- **SHA256 checksum:** e4316485cba75bf032cfcd4537d1c9281bf8813bac03d84004e55b5bf415ec99

**NOTE****How to generate a checksum**

- **MD5:** Use the following command:

```
curl -L -H 'Accept: application/vnd.github.v3.raw' -s https://$CONTRAST_GITHUB_PAT@maven.pkg.github.com/Contrast-Security-Inc/sast-local-scan-runner/com.contrastsecurity.sast-local-scan-runner/X.X.XX/sast-local-scan-runner-X.X.XX.jar.md5 -o sastXX.md5
```

- **SHA:** Use the following command:

```
curl -L -H 'Accept: application/vnd.github.v3.raw' -s https://$CONTRAST_GITHUB_PAT@maven.pkg.github.com/Contrast-Security-Inc/sast-local-scan-runner/com.contrastsecurity.sast-local-scan-runner/X.X.XX/sast-local-scan-runner-X.X.X.jar.sha1 -o sastXX.sha
```

For both types of checksums, replace `x.x.xx` with the version of the engine you are downloading and validating with a checksum. For example, for the 1.1.0 version of the engine, replace `x.x.xx` with `1.1.0`, and for the output (`SastXX.sha` or `MD5`), a value to represent the current version, such as `10`.

Application signing verification

To verify that Contrast created and signed the Scan local engine that you downloaded, use this command:

```
jarsigner -verify -verbose -certs sast-local-scan-runner-0.0.XX.jar
```

Replace `xx` with the version of the Scan local engine that you want to verify.

Scan local engine 1.1.4

Release date: November 26, 2024

New and improved:

- Added a new `--metadata` option for the Scan local engine that lets you specify metadata when you create a scan project.
- Added optional support for Rust and Terraform using the Semgrep open source engine. If you want to scan code in these languages and send the results to the Contrast web interface, you must download the Semgrep engine. If the Scan local engine identifies the presence of either of these languages, it sends the relevant files to Semgrep and combines the SARIF results with the file that Contrast creates.

[Scan languages with the Semgrep engine \(page 886\)](#) provides additional details about this feature.

Checksum:

- **MD5 checksum:** `fa99a209ba3662a198df735fa4c795eb`
- **SHA1 checksum:** `1c78f9570e20c18b01c4b609904f4bdf9cfe8eff`
- **SHA256 checksum:** `e4316485cba75bf032cfcd4537d1c9281bf8813bac03d84004e55b5bf415ec99`

**NOTE****How to generate a checksum**

- **MD5:** Use the following command:

```
curl -L -H 'Accept: application/vnd.github.v3.raw' -s https://$CONTRAST_GITHUB_PAT@maven.pkg.github.com/Contrast-Security-Inc/sast-local-scan-runner/com.contrastsecurity.sast-local-scan-runner/X.X.XX/sast-local-scan-runner-X.X.XX.jar.md5 -o sastXX.md5
```

- **SHA:** Use the following command:

```
curl -L -H 'Accept: application/vnd.github.v3.raw' -s https://$CONTRAST_GITHUB_PAT@maven.pkg.github.com/Contrast-Security-Inc/sast-local-scan-runner/com.contrastsecurity.sast-local-scan-runner/X.X.XX/sast-local-scan-runner-X.X.X.jar.sha1 -o sastXX.sha
```

For both types of checksums, replace `x.x.xx` with the version of the engine you are downloading and validating with a checksum. For example, for the 1.1.0 version of the engine, replace `x.x.xx` with `1.1.0`, and for the output (`SastXX.sha` or `MD5`), a value to represent the current version, such as `10`.

Application signing verification

To verify that Contrast created and signed the Scan local engine that you downloaded, use this command:

```
jarsigner -verify -verbose -certs sast-local-scan-runner-0.0.XX.jar
```

Replace `xx` with the version of the Scan local engine that you want to verify.

Scan local engine 1.1.3

Release date: October 8, 2024

New and improved:

- The `--memory` option now works with the binary scan engine.
Recommendation: Keep your memory allocation for the binary scan engine at 12GB or higher. A lower memory allocation can adversely affect the binary scan engine's performance and accuracy.
- To reduce noise and potential false positives, `.h` files are no longer implicitly scanned for C, C++, or ObjectiveC languages.
If, during the course of scanning these languages, the source code calls a `.h` file, then that file is scanned as part of the overall code analysis.

Checksum:

- **MD5 checksum:** `fa99a209ba3662a198df735fa4c795eb`
- **SHA1 checksum:** `1c78f9570e20c18b01c4b609904f4bdf9cfe8eff`
- **SHA256 checksum:** `e4316485cba75bf032cfcd4537d1c9281bf8813bac03d84004e55b5bf415ec99`

**NOTE****How to generate a checksum**

- **MD5:** Use the following command:

```
curl -L -H 'Accept: application/vnd.github.v3.raw' -s https://  
$CONTRAST_GITHUB_PAT@maven.pkg.github.com/Contrast-Security-  
Inc/sast-local-scan-runner/com.contrastsecurity.sast-local-  
scan-runner/X.X.XX/sast-local-scan-runner-X.X.XX.jar.md5 -o  
sastXX.md5
```

- **SHA:** Use the following command:

```
curl -L -H 'Accept: application/vnd.github.v3.raw' -s https://  
$CONTRAST_GITHUB_PAT@maven.pkg.github.com/Contrast-Security-  
Inc/sast-local-scan-runner/com.contrastsecurity.sast-local-  
scan-runner/X.X.XX/sast-local-scan-runner-X.X.X.jar.sha1 -o  
sastXX.sha
```

For both types of checksums, replace `X.X.XX` with the version of the engine you are downloading and validating with a checksum. For example, for the 1.1.0 version of the engine, replace `X.X.XX` with `1.1.0`, and for the output (`SastXX.sha` or `MD5`), a value to represent the current version, such as `10`.

Application signing verification

To verify that Contrast created and signed the Scan local engine that you downloaded, use this command:

```
jarsigner -verify -verbose -certs sast-local-scan-runner-0.0.XX.jar
```

Replace `xx` with the version of the Scan local engine that you want to verify.

Scan local engine 1.1.2

Release date: August 29, 2024

New and improved:

- Added a `--level` command option that provides better logging from the multi-language scan engine. To turn on logging for a specific log level, use one of these values: `ERROR`, `WARN`, `INFO`, `DEBUG`, or `TRACE`.
Use this option only when the Contrast Support team instructs you to do so rather than using it with all scans
- Changed the maximum log size to 20MB before creating a new log file.

Bug fixes:

- Due to a bug that caused the source code scan engine to fail, all customers should upgrade their local scanner to **version 1.1.2** to resume operations.
All previous versions are now considered end of life.
To help you understand the Contrast version control policy, [Scan local engine releases and versions \(page 875\)](#) describes the policy that applies to all future releases.

Checksum:

- **MD5 checksum:** 7be87ce1ab990c45e91c7060e5300ce2
- **SHA1 checksum:** e55d9fa9323dc93bc29d4f68e927763c6e5fb12b
- **SHA256 checksum:** ef8c84c1ad4549ab4e22a638dbf5d5d4d5700f6209ddcabfe66a20639880e0be



NOTE

How to generate a checksum

- **MD5:** Use the following command:

```
curl -L -H 'Accept: application/vnd.github.v3.raw' -s https://$CONTRAST_GITHUB_PAT@maven.pkg.github.com/Contrast-Security-Inc/sast-local-scan-runner/com.contrastsecurity.sast-local-scan-runner/X.X.XX/sast-local-scan-runner-X.X.XX.jar.md5 -o sastXX.md5
```

- **SHA:** Use the following command:

```
curl -L -H 'Accept: application/vnd.github.v3.raw' -s https://$CONTRAST_GITHUB_PAT@maven.pkg.github.com/Contrast-Security-Inc/sast-local-scan-runner/com.contrastsecurity.sast-local-scan-runner/X.X.XX/sast-local-scan-runner-X.X.X.jar.sha1 -o sastXX.sha
```

For both types of checksums, replace `x.x.xx` with the version of the engine you are downloading and validating with a checksum. For example, for the 1.1.0 version of the engine, replace `x.x.xx` with `1.1.0`, and for the output (`SastXX.sha` or `MD5`), a value to represent the current version, such as `10`.

Application signing verification

To verify that Contrast created and signed the Scan local engine that you downloaded, use this command:

```
jarsigner -verify -verbose -certs sast-local-scan-runner-0.0.XX.jar
```

Replace `xx` with the version of the Scan local engine that you want to verify.

Scan 1.1.1

Release date: June 14, 2024

New and improved:

- The Scan local engine now creates a unique output folder for each scan. The location of the folder name is: `.contrast-scan/<CURRENT_TIMESTAMP>`. where `<CURRENT_TIMESTAMP>` is the date and time when the scan ran.

Bug fixes:

- Updated configuration for C/C++ languages to avoid duplication of results.
In previous versions of the Scan local engine, scans analyzed `.h` and `.c` files using C++ and C rules. This behavior generated duplicate vulnerabilities. The latest version of the scan engine no longer generates duplicate vulnerabilities. If you had this issue previously, when you run the new version of the scan engine, it will change the status of the duplicates to **Remediated**.



IMPORTANT

The new multi-language source code scan engine is now version 1.1.1. Versions 1.0.0, 1.0.1, and 1.0.2, 1.0.5, and 1.0.6 are considered internal test and beta versions of the multi-language scan engine and are not available for download by Contrast customers.

Checksum:

- **MD5 checksum:** 4ad02dbb651afd65aa34540b74070460
- **SHA1 checksum:** 31fe66afb757422aab0cb9f59fc4f1d858146bce
- **SHA256 checksum:** 3f7fe7b9940c78b98721fdd865a058e0e3b61b65e45cd905615b91a828128ff7



NOTE

How to generate a checksum

- **MD5:** Use the following command:

```
curl -L -H 'Accept: application/vnd.github.v3.raw' -s https://$CONTRAST_GITHUB_PAT@maven.pkg.github.com/Contrast-Security-Inc/sast-local-scan-runner/com.contrastsecurity.sast-local-scan-runner/X.X.XX/sast-local-scan-runner-X.X.XX.jar.md5 -o sastXX.md5
```

- **SHA:** Use the following command:

```
curl -L -H 'Accept: application/vnd.github.v3.raw' -s https://$CONTRAST_GITHUB_PAT@maven.pkg.github.com/Contrast-Security-Inc/sast-local-scan-runner/com.contrastsecurity.sast-local-scan-runner/X.X.XX/sast-local-scan-runner-X.X.X.jar.sha1 -o sastXX.sha
```

For both types of checksums, replace `x.x.xx` with the version of the engine you are downloading and validating with a checksum. For example, for the 1.1.0 version of the engine, replace `x.x.xx` with `1.1.0`, and for the output (`SastXX.sha` or `MD5`), a value to represent the current version, such as `10`.

Application signing verification

To verify that Contrast created and signed the Scan local engine that you downloaded, use this command:

```
jarsigner -verify -verbose -certs sast-local-scan-runner-0.0.XX.jar
```

Replace `xx` with the version of the Scan local engine that you want to verify.

Scan 1.1.0

Release date: April 30, 2024

New and improved:

- Added these exclusions to the Java binary scanner: `com.azure`, `org.apache`, and `com.nimbusds`.
- Added a `--severity` parameter for the Scan local engine to let you get a build fail status. The specified values is the minimum level of severity that returns a build fail status code that you can use to gate builds in pipelines.
For example, if you specify `--severity high`, a finding of that severity or higher returns a build fail status code.
- Added support for multi-branch scanning when using the GitHub action for the Scan local engine.
- You can now [download the Scan local engine \(page 875\)](#) with a reusable script.

Bug fixes:

- Improved the Scan architecture to allow scanning of larger source code repos and faster processing of a large amount of findings.

**IMPORTANT**

The new multi-language source code scan engine is now version 1.1.0. Versions 1.0.0, 1.0.1, and 1.0.2, 1.0.5, and 1.0.6 are considered internal test and beta versions of the multi-language scan engine and are not available for download by Contrast customers.

Checksum:

- **MD5 checksum:** 4ad02dbb651afd65aa34540b74070460
- **SHA1 checksum:** 31fe66afb757422aab0cb9f59fc4f1d858146bce
- **SHA256 checksum:** 3f7fe7b9940c78b98721fdd865a058e0e3b61b65e45cd905615b91a828128ff7

**NOTE****How to generate a checksum**

- **MD5:** Use the following command:

```
curl -L -H 'Accept: application/vnd.github.v3.raw' -s https://  
$CONTRAST_GITHUB_PAT@maven.pkg.github.com/Contrast-Security-  
Inc/sast-local-scan-runner/com.contrastsecurity.sast-local-  
scan-runner/X.X.XX/sast-local-scan-runner-X.X.XX.jar.md5 -o  
sastXX.md5
```

- **SHA:** Use the following command:

```
curl -L -H 'Accept: application/vnd.github.v3.raw' -s https://  
$CONTRAST_GITHUB_PAT@maven.pkg.github.com/Contrast-Security-  
Inc/sast-local-scan-runner/com.contrastsecurity.sast-local-  
scan-runner/X.X.XX/sast-local-scan-runner-X.X.X.jar.shal -o  
sastXX.sha
```

For both types of checksums, replace `X.X.XX` with the version of the engine you are downloading and validating with a checksum. For example, for the 1.1.0 version of the engine, replace `X.X.XX` with `1.1.0`, and for the output (`SastXX.sha` or `MD5`), a value to represent the current version, such as `10`.

Application signing verification

To verify that Contrast created and signed the Scan local engine that you downloaded, use this command:

```
jarsigner -verify -verbose -certs sast-local-scan-runner-0.0.XX.jar
```

Replace `xx` with the version of the Scan local engine that you want to verify.

Scan 1.0.9

Release date: March 15, 2024



NOTE

This version of the Scan local engine is available by request only. Contrast is not publishing checksum information at this time.

To request access to this version of the Scan local engine, follow your normal Contrast support process.

Contrast plans to make the new Scan local engine generally available in the near future.

New and improved:

- Added a `--timeout` CLI option that lets you control the maximum time the multi-language source code scan engine scans the specified source code.
The value for this option is a specified number of minutes. This option applies to each language. For example, if you set the value of this option to 120 minutes and your repo contains four languages, potentially, the scan can take up to eight hours (120 minutes x 4 languages).
This feature is only available for the Contrast local scan engine only.
- Added support for file and folder exclusions.
To use this feature, add a file named `.contrast-scan.json` to the root folder of the source code you are going to scan. [Exclude files and folders \(page 882\)](#) describes how to use this feature.
This feature is only available for the Contrast local scan engine and is only supported for multi-language source code scans.
The file format for the JSON file is:

```
// File name ".contrast-scan.json"
{
  "excludes": [
    "**/MavenWrapperDownloader.java",
    "**/*.js"
  ]
}
```

- Scans automatically fail if the multi-language source code scan engine doesn't find any technologies in the submitted code.

Bug fixes:

- Fixed a bug that could cause a race condition, resulting in slow performance.
- Fixed a bug that caused incorrect date formats to be generated in the SARIF output. The incorrect formats caused which caused errors when using the SARIF output in Github.

**IMPORTANT**

The new multi-language source code scan engine is now version 1.0.9. Versions 1.0.0, 1.0.1, and 1.0.2, 1.0.5, and 1.0.6 are considered internal test and beta versions of the multi-language scan engine and are not available for download by Contrast customers.

Application signing verification

To verify that Contrast created and signed the local scan engine that you downloaded, use this command:

```
jarsigner -verify -verbose -certs sast-local-scan-runner-0.0.XX.jar
```

Replace `xx` with the version of the local scan engine that you want to verify.

Scan 1.0.8

Release date: February 15, 2024

**NOTE**

This version of the local scan engine is available by request only. Contrast is not publishing checksum information at this time.

To request access to this version of the local scan engine, follow your normal Contrast support process.

Contrast plans to make the new local scan engine generally available in the near future.

New and improved:

- Added support for scanning in the repo for Github customers.
Starting with 1.0.8, Scan supports a new Github action that supports main branch scanning in a Github repo. This feature supports failing builds based on the presence of a specified vulnerability severity (or higher). Learn more at [Use Contrast Scan with GitHub repositories \(page 884\)](#).
- Increased the minimum memory requirement for the multi-language scan engine to 8 GB and the timeout setting to 60 minutes. This does not replace the minimum memory requirement of 12 GB when scanning .JAR and .WAR files using the Java binary scanner. We continue to recommend that all users of the local scan engine should ensure that 12 GB of memory is available when running scans.

Bug fixes:

- Addressed a number of issues that prevented some languages from being correctly identified by the multi-language source code scan engine when scanned by the local scan engine. All languages identified by the multi-language source code scan engine should now correctly identify and be scanned.

**IMPORTANT**

The new multi-language source code scan engine is now version 1.0.8. Versions 1.0.0, 1.0.1, and 1.0.2, 1.0.5, and 1.0.6 are considered internal test and beta versions of the multi-language scan engine and are not available for download by Contrast customers.

Application signing verification

To verify that Contrast created and signed the local scan engine that you downloaded, use this command:

```
jarsigner -verify -verbose -certs sast-local-scan-runner-0.0.XX.jar
```

Replace `xx` with the version of the local scan engine that you want to verify.

Scan 1.0.7

Release date: January 25, 2024

**NOTE**

This version of the local scan engine is available by request only. Contrast is not publishing checksum information at this time.

To request access to this version of the local scan engine, follow your normal Contrast support process.

Contrast plans to make the new local scan engine generally available in the near future.

New and improved:

- Increased the memory that the multi-language source code scan engine uses to 2G to better support larger code bases. The minimum memory requirement when using the local scan engine is still 12GB.
- Added a `--memory` parameter to the CLI that you can use to override the allocated memory for the multi-language source code scan engine.
- Added additional logging to capture the parameters used when invoking the local scan engine. This logging captures the entire invocation command for the local scan engine (for example, `-r`, `-p` and so forth) for use when troubleshooting errors.

**IMPORTANT**

The new multi-language source code scan engine is now version 1.0.7. Versions 1.0.0, 1.0.1, and 1.0.2, 1.0.5, and 1.0.6 are considered internal test and beta versions of the multi-language scan engine and are not available for download by Contrast customers.

Bug fixes:

- Addressed an issue when scanning .NET applications that resulted in source code being incorrectly identified
- Addressed an issue that caused the multi-language scan engine to ignore ABAP code when presented in a code artifact

Application signing verification

To verify that Contrast created and signed the local scan engine that you downloaded, use this command:

```
jarsigner -verify -verbose -certs sast-local-scan-runner-0.0.XX.jar
```

Replace **xx** with the version of the local scan engine that you want to verify.

Scan 1.0.4

Release date: December 14, 2023



NOTE

This version of the local scan engine is available by request only. Contrast is not publishing checksum information at this time.

To request access to this version of the local scan engine, follow your normal Contrast support process.

Contrast plans to make the new local scan engine generally available in the near future.

Bug fixes:

- Fixed a bug that prevented VB.NET and Scala source code from being correctly identified and scanned by the multi-language engine.



IMPORTANT

The new multi-language source code scan engine is now version 1.0.4. Versions 1.0.0, 1.0.1, and 1.0.2 are considered internal test and beta versions of the multi-language scan engine and are not available for download for Contrast customers.

Application signing verification

To verify that Contrast created and signed the local scan engine that you downloaded, use this command:

```
jarsigner -verify -verbose -certs sast-local-scan-runner-0.0.XX.jar
```

Replace **xx** with the version of the local scan engine that you want to verify.

Scan 1.0.3

Release date: November 2023

**NOTE**

Local Scan Engine 1.0.3 is currently on restricted release. As a result, we are not providing checksum information at this time.

To get access to this version, open a [support ticket](#) to request it. We apologize for this inconvenience and are working hard to address this issue as soon as possible.

New and improved:

November 29, 2023

- Fixed an issue in role-based access control authentication that could trigger a 403 error when you try to assign a project to an empty resource group or when a user has access to multiple resource groups and they do not specify one.
If role-based access control is turned on, the `-r <ResourceGroupName>` option in the Contrast CLI is now mandatory when you create a scan project.

November 8, 2023

- The Contrast local scan engine now supports the ability to scan source code for over 25 languages. For a complete list of supported languages, see [Contrast Scan supported languages \(page 657\)](#).
- The local scan engine can now run natively under Windows environments running a suitable JVM.
- Fixed an issue where using spaces in the path for an artifact to be scanned caused a fatal scan error.
- Removed an unneeded log from the local scan engine, reducing overall disk space utilization when scanning Java binary files (JAR or WAR files).
- Fixed an issue that caused the local scan engine to fail when running under Alpine Linux.

**IMPORTANT**

The new multi-language source code scan engine is now version 1.0.3. Versions 1.0.0, 1.0.1, and 1.0.2 are considered internal test and beta versions of the multi-language scan engine and are not available for download for Contrast customers.

Application signing verification

To verify that Contrast created and signed the local scan engine that you downloaded, use this command:

```
jarsigner -verify -verbose -certs sast-local-scan-runner-0.0.XX.jar
```

Replace `xx` with the version of the local scan engine that you want to verify.

Scan 0.0.63

Release date: July 24, 2023

Bug fixes:

- Fixed a bug that prevented the local scanner from reporting all vulnerabilities found across multiple JAR files. Only the last JAR file scanned in the ZIP file was reported.

Checksum:

- **MD5 checksum:** f57f9174d0643832f9e38b95998fe280
- **SHA checksum:** 8b2f5680111c5a4e5999a3449ee871bb822d27f6

**NOTE****How to generate a checksum**

- **MD5:** Use the following command:

```
curl -L -H 'Accept: application/vnd.github.v3.raw' -s https://$CONTRAST_GITHUB_PAT@maven.pkg.github.com/Contrast-Security-Inc/sast-local-scan-runner/com.contrastsecurity.sast-local-scan-runner/X.X.XX/sast-local-scan-runner-X.X.XX.jar.md5 -o sastXX.md5
```

- **SHA:** Use the following command:

```
curl -L -H 'Accept: application/vnd.github.v3.raw' -s https://$CONTRAST_GITHUB_PAT@maven.pkg.github.com/Contrast-Security-Inc/sast-local-scan-runner/com.contrastsecurity.sast-local-scan-runner/X.X.XX/sast-local-scan-runner-X.X.X.jar.shal -o sastXX.sha
```

For both types of checksums, replace `x.x.xx` with the version of the engine you are downloading and validating with a checksum. For example, for the 0.0.60 version of the engine, replace `x.x.xx` with `0.0.60`, and for the output (`SastXX.sha` or `MD5`), a value to represent the current version, such as `60`.

Application signing verification

To verify that Contrast created and signed the local scan engine that you downloaded, use this command:

```
jarsigner -verify -verbose -certs sast-local-scan-runner-0.0.XX.jar
```

Replace `xx` with the version of the local scan engine that you want to verify.

Scan 0.0.60

Release date: May 22, 2023

New and improved:

- Added the ability to specify a resource group as a parameter in the local scan engine when you scan a project for the first time.
To use this feature, your organization must have role-based access control enabled and you require sufficient permissions to create a new project (Manage Project Role or higher).
Specify the resource group name using the `-r` parameter.

Checksum:

- **MD5 checksum:** 0fa38c5c9e46e3b2c6bdb2d2ed3baa20
- **SHA checksum:** 76fe00f7d70d45176904a2b62a9d1083f0731a03

**NOTE****How to generate a checksum**

- **MD5:** Use the following command:

```
curl -L -H 'Accept: application/vnd.github.v3.raw' -s https://  
$CONTRAST_GITHUB_PAT@maven.pkg.github.com/Contrast-Security-  
Inc/sast-local-scan-runner/com.contrastsecurity.sast-local-  
scan-runner/X.X.XX/sast-local-scan-runner-X.X.XX.jar.md5 -o  
sastXX.md5
```

- **SHA:** Use the following command:

```
curl -L -H 'Accept: application/vnd.github.v3.raw' -s https://  
$CONTRAST_GITHUB_PAT@maven.pkg.github.com/Contrast-Security-  
Inc/sast-local-scan-runner/com.contrastsecurity.sast-local-  
scan-runner/X.X.XX/sast-local-scan-runner-X.X.X.jar.shal -o  
sastXX.sha
```

For both types of checksums, replace `x.x.xx` with the version of the engine you are downloading and validating with a checksum. For example, for the 0.0.60 version of the engine, replace `x.x.xx` with `0.0.60`, and for the output (`SastXX.sha` or `MD5`), a value to represent the current version, such as `60`.

Application signing verification

To verify that Contrast created and signed the local scan engine that you downloaded, use this command:

```
jarsigner -verify -verbose -certs sast-local-scan-runner-0.0.XX.jar
```

Replace `xx` with the version of the local scan engine that you want to verify.

Scan 0.0.56 - 0.0.59

Release date: April 6, 2023

New and improved:

- Support for multi-JAR scanning

This release adds the ability to scan multiple JAR files as one artifact. You can add multiple JAR files to a ZIP file and scan it as a single artifact.

To scan a multi-JAR ZIP file, package the JAR files at the top level in a ZIP file and scan it using the Scan local engine, as normal. For example:

```
multiple-jar-artifact.zip  
-> artifact1.jar  
-> artifact2.jar  
-> artifact3.jar
```

Once completed, the Contrast web interface displays the scan as a single project under the Scans tab.

Bug fixes:

Releases 0.0.57 through 0.0.59 contained internal bug fixes that had no effect on the Scan behavior or performance.

Scan web interface

April 2025 Scan web interface

Release date: April 15, 2025

New and improved:

- Added a visual representation of total lines of code scanned for individual languages.
- Added a Created time column to the Scan project list that shows the date and time when you created a scan project.
This feature applies to new scans only.
- Added a new tab on the Scan project page that shows files and rule that were excluded based on file exclusions and policies.
This tab is available only for scans with the Contrast Scan local engine.
- The Contrast web interface now shows a warning message when you upload a code artifact to be scanned that navigating away before the upload completes cancels the upload operation.
- Added support to configure notifications for failed scans in the Contrast web interface.

March 2025 Scan web interface

Release date: March 18, 2025

New and improved:

- Added the ability to see the total number of lines of code scanned during a particular scan.
- Added the ability to see the total number of files scanned during a particular scan.
- Added a visual representation of total lines of code scanned for individual languages.
- Added the ability to see the version of the local scanner used during a scan.
- Added a new tab on the Scan project page that shows files that were excluded based on file exclusions and policies.

February 2025 Scan web interface

Release date: February 20, 2025

New and improved:

- Adjusted the default sorting of projects.
By default, projects are now automatically sorted by last scan date in ascending order (the newest project is at the top, the oldest is at the bottom).
- Added a 90 day option to the Last Scan filter on the Scan project page.
- For organizations with role-based access control enabled, the Scan project page now shows the resource groups with which the project is associated.
The displayed resource groups are based on the role for the logged-in user. Different users could see different resource groups. For example, a standard user could see different resource groups than a user with administrator permissions.
- Added a Branch tab to the Scan project page.
This tab is visible only when you run scans using the branch command with the Scan local engine.

January 2025 Scan web interface

Release date: January 21, 2025

New and improved:

- Added more details about the source of a vulnerability.
Where relevant, the vulnerability overview displays the source file name, line number and code snippet as well as the sink file name, line number and code snippet.

- Added more details for the reason a scan failure.
In the Scan history, you can hover over a scan fail message to see a tool tip that displays information on the reason for a scan failure.

Archive

January 2024 - December 2024

December 2024 Scan web interface

Release date: December 10, 2024

New and improved:

- Added support for dynamic scoring when you change the status of a vulnerability to **Not a Problem**.

Bug fixes:

- Fixed an issue that prevented the display of metadata values as tags for a scan project.

November 2024 Scan web interface

Release date: November 26, 2024

New and improved:

- Added support for metadata for scan projects.
Metadata consists of a key-value pair that is displayed on the Scan project page. You have the option of restricting scan project creation when required metadata is missing.
Currently, this feature is available for new scan projects only.
- For enhanced clarity, the Scan vulnerability list now shows the names of source files and line numbers.

Bug fixes:

- Fixed a bug that prevented the Language column on the Scan project page from correctly reflecting in scope languages when you selected other filters.
- Fixed a bug that caused a scan to appear as though it failed due to overly large code snippets being present in the vulnerability overview.

October 2024 Scan web interface

Release date: October 8, 2024 (updated October 17, 2024)

New and improved:

- When you download a CSV file that contains more than 2,000 vulnerabilities, you are now prompted to select individual pages of results that contain up to 2,000 vulnerabilities. For example, if a report contains 5,400 vulnerabilities, when you download it, you have the option of selecting Page 1, Page 2, or Page 3. You download each page individually and can combine them afterwards.
Selecting multiple pages is not supported.
- Increased the number of vulnerabilities included in the Attestation report from 100 to 3,000.
- Increased the number of vulnerabilities included in the General Vulnerability report to 3,000
- Added a new column to the Vulnerabilities tab that shows the CWE for the vulnerability. This column can has a filter you can use to refine the view.
- Added the ability for a user with the View, edit, and delete project action (role-based access control) or Organization Admin (organization users and groups access control) to change the severity of a detected vulnerability.

To change the severity of a vulnerability, select the current severity and select an option from the dropdown (for example, select High and change it to Critical or Medium).

If more than one vulnerability of the same type exists, you have the option of changing the severity for the selected vulnerability only or the severity of all matching vulnerabilities. Future scans do not override this severity change.

- Added a Secure Code Warrior Guidelines tab to an individual vulnerability. This tab uses the associated CWE for a particular vulnerability to provide Secure Code warrior guidelines and training video information. The purpose of this information is to provide additional context on the vulnerability and ways to resolve it.

Where possible, the guidelines reflect the vulnerability language. If the CWE does not support that language is not supported, the tab displays generic guidelines. If no guidelines or information exists for a specific CWE, the tab is not available.

September 2024 Scan web interface

Release date: September 10, 2024

New and improved:

- Fixed an issue where changing the status of a vulnerability to **Not a Problem** could change to **Remediated** if subsequent scans didn't discover the vulnerability. Now, the status of **Not a Problem** never changes.
To have vulnerabilities assessed again, change the status to **Confirmed** or **Suspicious**.
- You can now change the status of a single vulnerability and apply that change to all vulnerabilities of the same type.

August 2024 release: Scan web interface

Release date: August 29, 2024

New and improved:

- Added the ability to create tags for Scan projects.
[Add tags to Scan projects \(page 662\)](#) describes how to use this feature.
- **NEW:** General Vulnerability report: a PDF report based on the CSV report.
This report includes the first 3,000 open vulnerabilities in the project based on severity and status.
- Added the ability to change the status for all vulnerabilities of the same type simultaneously.
If you update the status of a large number of vulnerabilities (1,000 or more) at one time, this change can take several minutes to complete. Contrast displays a message in the web interface when this action is done.
- Added the ability to use the last Contrast Scan date to filter and sort scan projects.

July 2024 release: Scan web interface

Release date: July 16, 2024

New and improved:

- Added the ability to update the status of multiple vulnerabilities at the same time, as described in [Batch edit Scan vulnerability status. \(page 895\)](#)

Bug fixes:

- Fixed an issue that prevented some how-to-fix information from displaying correctly for VB.NET and ABAP vulnerabilities

June 2024 release: Scan web interface

Release date: June 11, 2024

New and improved:

- Added the ability to see the language associated with a detected vulnerability.
To see content in the new Language column, run a new scan in the project. The Contrast web interface doesn't display the language for older scans.
If you are using the Scan local engine, you must use version 1.1.1. Using an earlier version of the local scan engine results in the Contrast web interface displaying **Composite** as the language. If you see this and you are using the Scan local engine, upgrade to version 1.1.1.
- Added the ability to filter views based on the language associated with a detected vulnerability.

Bug fixes:

- Fixed a bug that prevented the How-to-fix information from being properly displayed in the Web interface.
- Fixed an issue that caused C++ and C# vulnerabilities to be counted twice.
As a result of this change, the system remediation workflow marks duplicate vulnerabilities as remediated for C++. For C#, these vulnerabilities remain open.

May 2024 release: Scan web interface

Release date: May 14, 2024

Bug fixes:

- Due to performance issues when generating the Scan attestation report, the report is now limited to generating 100 open vulnerabilities, by severity and status. Larger reports will be supported in the future.
- Addressed an issue with file uploads to Contrast where files over 500 MB could cause out of memory (OOM) errors, especially when you used the Scan CLI commands. . This fix does not increase the file upload size beyond 1GB but provides a consistent user experience between uploads to the Contrast web interface and with the CLI. If you have repos that are larger than 1 GB, consider using the Contrast Scan local engine.

April release: Scan web interface

Release date: April 25, 2024

New and improved:

- Enhanced the CSV report to include code snippets for each vulnerability.
- Changed the CSV report so that you can see the file name and line number from the path for each vulnerability.
- Changed the CSV report to exclude vulnerabilities with a **Remediated** and **Not a problem** status.
- Added the ability to supply a filter to the API call when generating a CSV report programmatically.
- To aid in timely generation of the CSV report and address performance issues, the CSV report is now limited to the first 2,000 open entries based on severity and status.
- Added pagination to the API when generating a CSV report so it can exceed the 2,000 line limit.

Bug fixes:

- Improved the Scan architecture to allow scanning of larger source code repos and faster processing of a large amount of findings.

March release: Scan web interface

Release date: March 2024

Bug fixes:

- Fixed a bug that could cause a race condition, resulting in slow performance in the Contrast web interface.
- Fixed a bug in the Contrast web interface that resulted in an error when specifying an underscore (_) as part of a search parameter when searching projects.

January release: Scan web interface and CLI

Release date: January 2024

New and improved:

- January 25, 2024

New and improved;

- On the Scan project page in the Contrast web interface, you can view the languages that the multi-language source code scan engine detected.
- Added the ability to search for scan projects based on detected languages.

Bug fixes:

- Fixed a bug in the CLI that suggested a scan had failed when it invoked the multi-language source code scan engine.
- Fixed a bug in the CLI that prevented the list of found vulnerabilities from being displayed in the CLI output once a scan completes.
- Addressed an issue when scanning .NET applications that resulted in source code being incorrectly identified
- Addressed an issue that caused the multi-language scan engine to ignore ABAP code when presented in a code artifact

January - December 2023**December release: Scan web interface**

Release date: December 2023

New and improved:

- December 14, 2023
 - **NEW:** You can now generate an Attestation report for your scan projects from a scan project page and from the vulnerability tab on the scan project page.
 - Removed the ability for a user to change a vulnerability status to **Fixed**. The Scan engine determines this status based on whether a vulnerability is still seen in the source code in subsequent scans.
 - Fixed a bug that prevented VB.NET and Scala source code from being correctly identified and scanned by the multi-language engine.

November release: Scan web interface

Release date: November 2023

New and improved:

- November 28, 2023
 - If role-based access control is turned on, creating a scan project now requires that you specify a resource group. The Create your scan project screen has a dropdown that displays a list of the resource groups assigned to the user who is creating the project. If users have a single resource group assigned to their role, this resource group is the default selection. In addition, users need a role that includes the **Create project** action. [Create a scan project \(page 665\)](#) describes this new requirement.

- November 8, 2023
 - **NEW:** Contrast Scan now provides two types of scans: Java binary for Java files, and source code for most other languages and technologies.
When you select a source code scan, upload a ZIP file that contains the source code you want to scan.
 - **NEW:** Source code scanning is expanded to include over 25 additional languages and technologies, as listed in [Scan supported languages and technologies \(page 657\)](#). To use the expanded source code scanner, select the **Source code** option when you create a new project.
 - **For hosted customers:** Contrast Scan now supports multi-language detection for source code scanning. When you upload a ZIP file, the scan engine determines which languages are present in the ZIP file and scans each file. Contrast displays the results in a single scan project.
 - Removed the need to select a language when you create a scan project. Scan can now determine the type of code artifact you are uploading. Scan continues to support single JAR and WAR files as well as ZIP files that contain multiple JAR files or source code.
 - Added two fields to the CSV file you can download:
 - **Language:** Identifies the language for a specific vulnerability.
 - **Comment:** Shows the last comment made for a vulnerability.The CSV file populates these fields after you run a new scan for an existing project.

June release: Scan web interface

Release date: June 2023

New and improved

- June 30, 2023
 - Added the ability to add a comment for a vulnerability status without changing the current status. The Activity tab for a specific vulnerability lets you add comments.
- June 12, 2023
 - Added the ability to see who created a project by displaying the project creator's name at the top of the Scans page and the Scan details page.
 - Added the ability to see who ran a specific scan for a project.
The Scan history in the Scans page has a new Name column that shows the name of the individual who ran a specific scan. The Summary section of the Scan details page also shows who ran the scan.



NOTE

Both of these features apply to new projects and new scans. Existing projects or scans do not display the new information.

May release: Scan web interface

Release date: May 2023

New and improved:

- Added support for multi-JAR scanning in the Java binary scanner.
You can now include multiple JAR files in a single ZIP file when you use the hosted Java binary scanner (using the Contrast CLI or the Contrast web interface).
The maximum upload size limit for a ZIP file is 1 GB.

April release: Scan web interface

Release date: April 2023

New and improved:

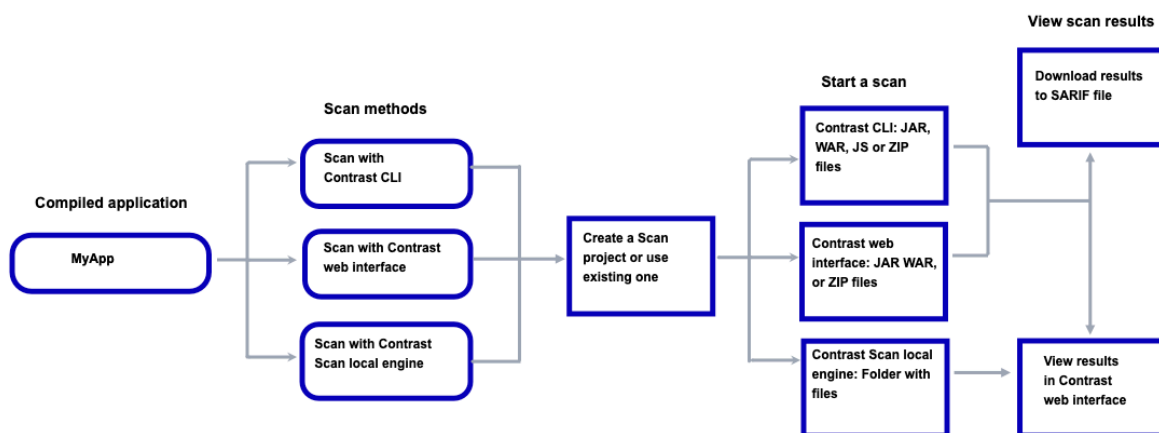
- Added a vulnerability activity tab that shows information on status changes made to vulnerabilities within a project.
To view this tab, select the Vulnerabilities tab for selected scan project and then, select a specific vulnerability
- Added the requirement to add comments when you change the status of a vulnerability in a project.
- Added the ability to delete a project and all associated data in the Contrast web interface for users with a Manage all projects role.

Scan process

This section provides details about the workflow for using Contrast Scan as well as the process Scan uses when analyzing your code.

Scan workflow

This diagram illustrates the Scan workflow that you follow.



Data entry points for Java binary scans

When scanning JAR files (compiled java binaries), the Java binary scan engine looks for data entry points typically found in web applications to find code to scan. If the compiled application does not contain data entry points, the scan completes successfully, but does not find vulnerabilities. For these scenarios, scan the uncompiled source code instead. If you previously scanned the JAR binary, create a new project and then, scan the uncompiled source code.

Here are some examples of typical data entry points that the Java binary scan examines:

pet Everything about your Pets		
POST	/pet/{petId}/uploadImage	uploads an image
POST	/pet	Add a new pet to the store
PUT	/pet	Update an existing pet
GET	/pet/findByStatus	Finds Pets by status
GET	/pet/findByTags	Finds Pets by tags
GET	/pet/{petId}	Find pet by ID
POST	/pet/{petId}	Updates a pet in the store with form data
DELETE	/pet/{petId}	Deletes a pet

Contrast Scan supported languages and technologies

Contrast Scan supports these languages and technologies:

Technology	Latest supported version	Associated file extensions	Artifacts to upload
Java binary scan			
Java (for example: J2EE, JSP, and Spring MVC)	Oracle Java 17	java	JAR and WAR files or a ZIP file with JAR or WAR files in the root directory of the ZIP file.
Source code scan			
ABAP	7.51	abap ,bsp, asprog, aclass, aint, asfinc, asfugr, appl, component	ZIP files or folders that contain files to scan. Contrast Scan automatically detects the language of the files.
ActionScript	3	as	
ASP.NET	Current version	asax, ascx, ashx, asmx, aspx, master	
C#	9 and later	cs, cshtml	
C	18	c, h, pc	
C++	20	h, hh, cpp, hpp, cc, pc	
COBOL	Current version	cob, cbl, cpy, pco	
Go	1.13	go	
Hana SQL Script	Current version	sql	
HTML	Current version	htm, html, xhtml	
Informix	Current version	sql, 4gl	

Technology	Latest supported version	Associated file extensions	Artifacts to upload
Java	Oracle Java 17 and later Contrast Scan can scan Oracle Java LTS 21 code, however any specific Oracle Java 21 controls are not scanned. The Contrast scan engine is not fully Oracle Java 21 LTS compliant.	java	
JavaScript/TypeScript	ES5	js, xsjs, ts, tsx	
JCL	Current version	jcl,prc	
JSP	Current version	jsp,j spx, xhtml	
Kotlin	1.6	kt, kts, ktm	
NATURAL	Current version	nls, nlp, nlh, nlm, nss, nsp, nsh	
Objective-C	2	h, m	
Oracle Forms	Current version	oforms	
PHP	7.4	php,php3,php4,php5,php6,phps,phtml	
PL-SQL	Current version	sql, sf, sps, spb, sp, fnc, spp, plsqli, trg, st, prc, pks, pkb, pck	
PowerScript	11.5	sru, sra, srw, srf, srs, srm, srx	
Python	3.9	python, py	
Ruby		rb	
RPG4	7.4	rpg, rpg3, rpg4, rpgle, dspf, mbr	
Scala	2.13	scala	
Swift	5.3	swift	
Transact-SQL	Current version	sql, tsqli,sp	
TypeScript	Current version	js,xsjs,ts,tsx	
Visual Basic 6	Current version	bas,frm,cls	
VB.NET	14	vb	
XML	Current version	xml	

Limited support with Semgrep open source engine

Only the Scan local engine supports these languages.

Language	Details
Terraform	Visit semgrep-rules/terraform at develop · semgrep/semgrep-rules
Rust	Visit rust/lang/security
Ruby 3.X	Visit semgrep.dev/p/ruby

Semgrep and associated rules are copyrighted software made available under Version 2.1 of the GNU Lesser General Public License. Complete source code for Semgrep, including complete copyright information, is located [here](#).

Visit the links in the table for complete source code for rules. You can access and update them in the Contrast Scan local engine JAR file using the information in [Create custom Scan rule exclusions \(page 674\)](#).

Contrast provides scanning of Terraform and Rust as is. [Scan languages with the Semgrep engine \(page 886\)](#) provides details about using the Semgrep open source scanner with the Contrast Scan local engine.

Scan package preparation

To get the best results from a scan, consider these best practices before you upload packages.



NOTE

This topic applies to the use of [hosted scans, \(page 668\)](#) where you upload files to Contrast. [Package preparation for local scans \(page 672\)](#) describes the best practices for the Scan local engine.

Artifact types

- For Java binary scans, upload either a WAR or JAR package.
- For source code scans, upload a ZIP package that contains the repository you want to scan.

You can include multiple JAR or WAR files in a ZIP package. Place these files at the root directory of the ZIP package, not in subdirectories.

The maximum uncompressed size of your code artifact must not exceed 1GB.

When preparing your ZIP file for uploading to Contrast, either with the CLI or the Contrast web interface, ensure the uncompressed data is no larger than 1 GB. Contrast Scan might reject uncompressed files that are larger than 1 GB. In that case, you need to split the files into multiple projects for successful scans.

Use a consistent file structure

Using a consistent file structure for each scan is crucial to preventing duplicate vulnerability findings. After an initial scan, if you need to change the file structure for the files you are scanning, create a new scan project for those files instead of using an existing one. If you are planning to scan multiple branches without using the [Contrast Scan Analyze](#) GitHub action, create a separate scan project for your personal branch.

If you use an existing scan project, change the file structure, and then run a scan, Contrast sets the original vulnerability status to **Remediated** and reports a new, duplicate vulnerability. The new finding is linked to the same file and line number but shows the new path.

Consistent file structure affects files in ZIP files and files **not** in a ZIP file.

Example 1: Changing ZIP file name

In this example, we rename a ZIP file but maintain the file structure in the ZIP file. This change doesn't result in Contrast reporting duplicate vulnerabilities:

```
scan.zip
|
|-- source_files

changed to

contrastscan.zip
```

```
|  
|-- source_files
```

In this case, the ZIP file name isn't part of the scan path, so changing it has no effect on the scan findings.

Example 2: Changing file structure in a ZIP file

In this example, we rename the ZIP file and also change the file structure by adding a new directory. This change causes Contrast to report duplicate vulnerabilities.

```
scan.zip  
|  
|-- source_files  
  
changed to  
  
contrastscan.zip  
|  
|-- contrastscan  
|  
|-- source files
```

In this case, changing the file structure within the ZIP file causes duplicate findings. To avoid this issue, create a new scan project and use it for future scans.

Example 3: Changing file structure in directories

In this example, we change the file structure by adding a new directory. This change causes Contrast to report duplicate vulnerabilities.

```
scan  
|  
|-- source_files  
  
changed to  
  
contrastscan  
|  
|-- contrastscan  
|  
|-- source files
```

In this case, changing the directory name and the file structure within the directory causes duplicate findings. To avoid this issue, create a new scan project and use it for future scans.

Access to class files and dependencies

If you package your files differently than suggested here, Scan has to make assumptions about your code. The results might not be as precise as they could be. They could include false negatives and positives.

When Scan has access to all the appropriate class files and dependencies, the results do not include phantom classes. A phantom class is a referenced class but either scan is unable to find bytecode for it or the scan was unable to decompile the code into intermediate representation (IR).

- Scan needs access to these files:
 - Application class files

- Application dependency jar or class files
- Organize application and dependencies in WAR files as described in the Oracle Java™ Servlet Specification.
- Organize applications and dependencies in JAR files similar to the way SpringBoot JAR files are organized.
SpringBoot JAR files place applications and dependencies in well-known areas.
- Including standard JDK files and common servlet container-provided dependencies are not required. Scan provides these dependencies for you.

Frameworks

To be able to deliver accurate results, Scan needs to understand the web framework that your application uses.

- **Source code scans:** This scan type supports all frameworks for the supported languages.
- **Java binary scans:** This scan type supports these frameworks:
 - Angular 8 or later
 - J2EE
 - Jakarta EE 2.0-3.0
 - jQuery
 - React 16 or later
 - SpringBoot
 - Spring MVC
 - Vue.JS 2 or later

Avoid use of thin JAR files

Thin JAR files contain only application byte code. These files require special execution loaders to dynamically access dependencies for loading. If you upload a thin JAR file, Scan does not execute any of your application code. It cannot access the application dependencies for accurate scanning.

View scan projects

The Scan project list shows these details for each scan project in your organization:

- **Score:** A letter grade that represents the potential security risk for an application based on the most recent scan in the project.
Scan uses the [Contrast application scoring \(page 1269\)](#).
- **Scan project:** The name of the scan project.
- **Vulnerable languages:** The languages in a project where Contrast detected vulnerabilities. The project can include multiple languages or just a single language.
- **Open vulnerabilities:** The number of open vulnerabilities that Contrast detected.
- **Last scan:** The amount of time since the last scan completed.
- **Created by:** The name of the person who created the scan project.
- **Created time:** The date and time when someone created the scan project.

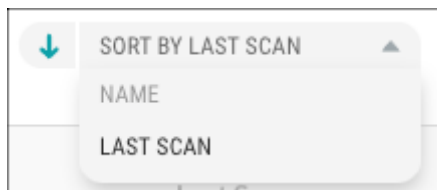
Before you begin

If role-based access control is turned on for your organization, verify that users have the correct [actions \(page 1278\)](#), [user access groups \(page 1278\)](#), and [resource groups \(page 1284\)](#) assigned to them.

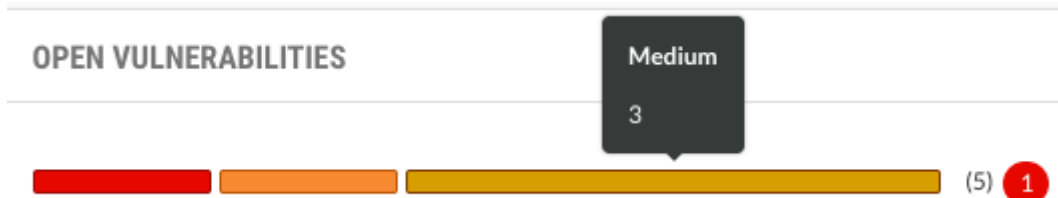
Steps

1. Select **Scans** in the header.
Contrast displays a list of scan projects.

2. To filter the view, select the small triangle (▾) at the top of the list and select a filter:
 - **All:** Shows all scan projects except for archived ones.
 - **Archived:** Shows only archived scan projects.
3. To find a specific scan project, select the magnifying glass icon (🔍) and enter a partial or full name of the project or a tag in the search box.
4. To sort the view, select the sort box and select an option:
 - **Name:** Sort the view by the project name.
 - **Last scan:** Sort the view based on when the the last scan completed.
By default, projects are automatically sorted by last scan date in ascending order (the newest project is at the top, the oldest is at the bottom).
 - **Created by:** Sort the view based on who created scan projects.
 Use the green arrow to sort by ascending or descending order.



5. To filter the list of projects, select the Filter icon (▾) next to a column heading:
 - a. **Scan project:** To view scans associated with specific tags, select one or more tags.
 - b. **Vulnerable languages:** To view only the projects that include specific vulnerable languages, select one or more languages.
 - c. **Last scan:** To filter the list according to the last scan time frame:
 - i. Select a specific time frame or select **Custom** to specify a preferred time frame.
 - ii. Select **Apply**.
 - d. **Created by:** To view scan projects according to the person who created them.
6. To view the number of vulnerabilities with a specific severity, hover over a section in the Vulnerabilities bar. To view vulnerabilities of a specific severity, select a section in the bar.



7. To view details for a specific project, select the project name in the Scan project column.

Add tags to scan projects

Use scan project tags to better organize scan projects and improve search functionality.

Before you begin

- **Users and groups:** An Organization Admin role is required.
- **Role-based access control (Preview):** A role with the Project Update action is required.
Role-based access control is part of the Contrast pre-release customer testing program. If you want to use this feature, contact your Contrast representative.

Steps

1. Select **Scans** in the header.
2. Select the checkbox on the left for one or more projects.

3. In the batch action menu at the bottom of the page, select the **Tag** icon (🔖).
4. Enter the name of a tag and select **Create tag**.

5. Create additional tags, as needed.
To remove an existing tag, select the **Delete** icon (X) next to it in the Tag project window.
6. Select **Save**.

Create metadata for scan projects

You can configure metadata that Contrast collects when you try to create a scan project. Configuring metadata lets you decide whether to restrict the creation of scan projects or running scans if metadata is missing.

The scan project list shows the configured metadata.

Before you begin

- If role-based access control is turned on, you need a role with the Manage organization action.
- If you are using organization users and groups for access control, you need the Organization Admin role.
- Currently, metadata is supported for new scan projects only.

Steps

1. From the user menu, select **Organization settings**.
2. Select **Scan projects**.
3. In Scan Configuration, for each field enter:
 - **Field type:** Freeform.
Freeform is the only supported field type.
 - **Value:** Enter a new value for this field.
 - **Value condition:** Use the checkbox to indicate whether the metadata value provided is **Required** or **Unique** for each project.
You can select both conditions.

The Scan local engine configuration preview shows the settings that you can include in a `.contrast-scan.json` file. If you want to use this file to configure the Scan local engine, create this file in the root folder of the source code you are scanning.

Organization Settings

Organization
Groups
Users
Access Control
Audit Log
Security
Agent Keys
Single Sign-On
Integrations
Servers
Applications
Notifications
Score Settings
Scan projects

Scan Configuration
Set key-value pairs

FIELD TYPE	VALUE	REQUIRED?	UNIQUE?
Freeform Text	TestingMetadata	<input checked="" type="checkbox"/> Required	<input type="checkbox"/> Unique value x

+ Add field

Scan local engine configuration [Learn more](#)

```
"metadata" : {  
  "testingMetadata" : "<value>"  
}
```

Restrict projects

☐ Restrict scan projects missing required fields

Cancel Save

4. Select **Add field** to complete as many rows as needed.
5. To prevent creation of scan projects or the ability to run a new scan that is missing all required fields, select **Restrict scan projects missing required fields**.
This restriction applies to new scan projects that you create with the Contrast web interface, the Contrast CLI, or the Contrast Scan local engine.
When you select this option, the Contrast web interface displays a warning message if you try to create a scan project without selecting the required metadata.

Configure dynamic scoring for Contrast Scan

Turning on dynamic scoring automatically adjusts the application score when you change the status for one or more scan vulnerabilities to **Not a Problem**. The application score no longer includes vulnerabilities with a status of **Not a Problem**.

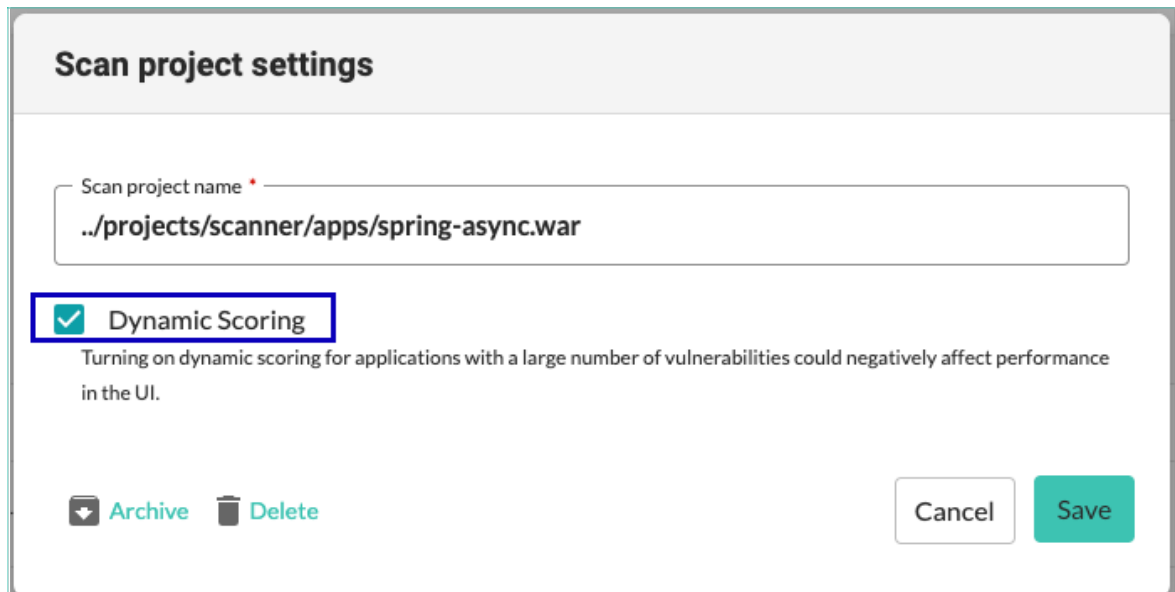
The default setting for this option is disabled.

Before you begin

- If role-based access control is turned on, you need a role with the Manage organization action.
- If you are using organizational users and groups, you need the Organization Admin role.
- Turning on dynamic scoring for applications with a large number of vulnerabilities can negatively affect performance in the Contrast web interface.

Steps

1. Select **Scans** in the header.
2. Select a scan project.
3. Select the **Settings** icon (⚙️) at the top of the list.
4. To turn on dynamic scoring, select the **Dynamic Scoring** checkbox.





Scan project settings

Scan project name *

../projects/scanner/apps/spring-async.war

☒ **Dynamic Scoring**

Turning on dynamic scoring for applications with a large number of vulnerabilities could negatively affect performance in the UI.

 Archive  Delete

Cancel Save

To turn off dynamic scoring, clear the checkbox.

5. Select **Save**.
6. In the Vulnerability tab for a scan project, select one or more vulnerabilities and change the status to **Not a Problem**.
7. To view the updated score, go to the Overview tab for the scan project.

See also

[Edit Scan vulnerability status \(page 894\)](#)

[Batch edit Scan vulnerability status \(page 895\)](#)

Create a scan project

Scan projects are containers for files that you want to scan.

Before you begin

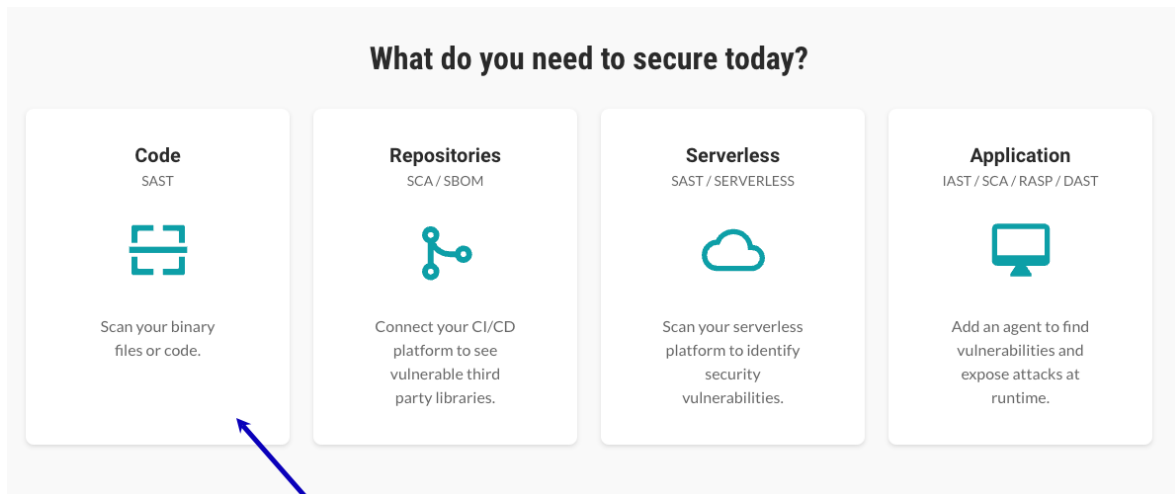
- Identify the files that you want to upload for scanning.
Scan supports different files types for each programming language. For example, for Java, upload a JAR or WAR file.
- If you are a hosted customer and role-based access control is turned on, your role must include the **Create project** action.
- If role-based access control is turned on for your organization, verify that users have the correct [actions \(page 1278\)](#), [user access groups \(page 1289\)](#), and [resource groups \(page 1284\)](#) assigned to them.

Steps

1. In the Contrast web interface, select **Add New** in the top right corner.



2. Select the **Code** card.



3. If you're a hosted customer and [role-based access control \(page 1277\)](#) is turned on, select a resource group from the dropdown.

Scan your code

Perform a SAST analysis of your code to find security risks.

- Resource Group *

All projects

This is the only resource group you have access to.
- Create your Scan project

Give your project a name.

[View scan projects](#)

Scan project name *

MyScanProject

Enter a minimum of 3 characters
- Include required metadata ⓘ

TestingMetadata *

TestingMetadata

Create Scan Project

Perform subsequent scans and see progress.

This container will allow you to perform scans of your code and provide progress on remediation efforts.

- If you have access to only one resource group, you don't select a resource group. Instead, you see the name of the resource group to which you have access.
 - If you're an on-premises customer or you are not using role-based access control, no resource group option is available.
4. Specify a name for the project.
Scan project names must be unique. Specify a name that lets you easily identify the scan project in other Contrast lists.
As a best practice, consider naming the project to match the name of the file. For example, if your file is `webgoat.jar`, name your project `webgoat` or `webgoat.jar`.
 5. Specify the values for the scan project metadata.
if metadata is required, you are unable to create the project until you specify the required metadata.

6. Select **Create scan project**.

**NOTE**

if you want to use an existing scan project instead of creating one, select **View scan projects**. This action opens the Scan project tab with your existing projects.

Next step

[Start a scan \(page 668\)](#)

Delete scan projects

If you want to permanently remove a scan project, you can delete it. Contrast permanently deletes all the data associated with the project.

You cannot undo this action.

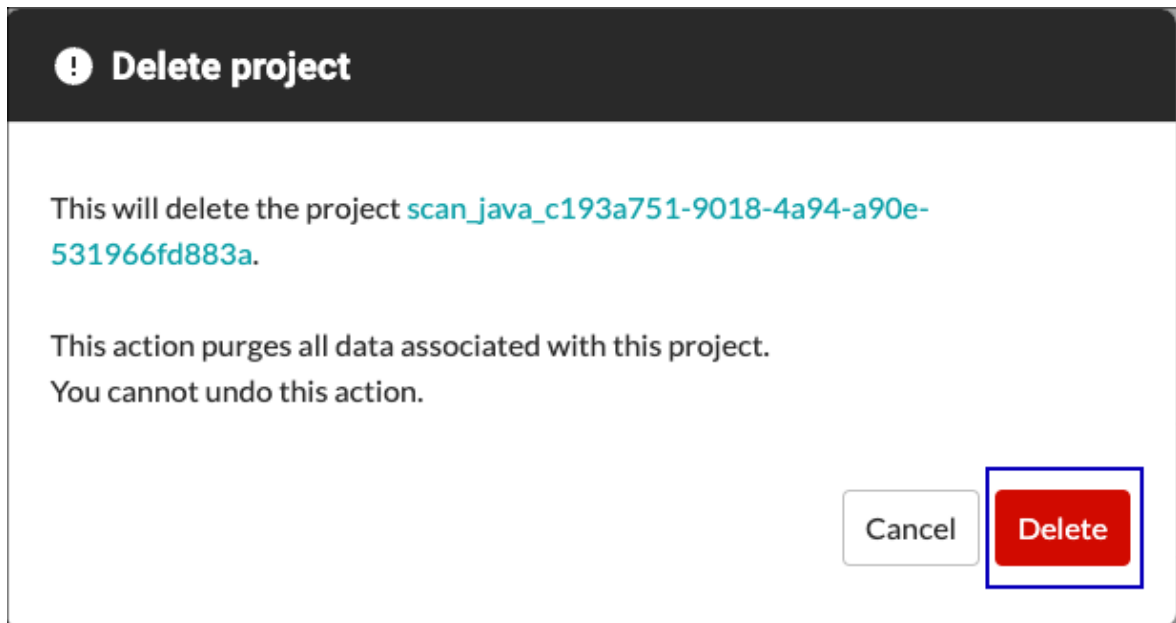
Before you begin

If role-based access control is turned on for your organization, verify that users have the correct [actions \(page 1278\)](#), [user access groups \(page 1289\)](#), and [resource groups \(page 1284\)](#) assigned to them.

Steps

1. Select **Scans** in the header.
2. Select a scan project.
3. Select the Settings icon (⚙️).
4. In the Scans projects settings window, select **Delete**.

5. In the Delete project window, confirm that you want to delete the project by selecting **Delete**.



Start a scan

Start a scan when you want to do the following tasks:

- Begin analysis of a new application.
- Test code changes you made to fix vulnerabilities on an application you scanned previously.

Before you begin

- Identify the scan project that contains the file you want to scan again or [create one \(page 665\)](#).
- If you navigate away from the page while an upload is in progress, Contrast Scan cancels the upload. Contrast displays a warning message if you try to leave the page.

Steps

1. Select **Scans** in the header.
2. Select a scan project.
3. Select **New Scan**.



4. From the displayed window, select a file or folder to upload and select **Open** or **Upload**. If the scan project contains a previous scan, select a new version of the file you previously scanned. The scan starts automatically after the file upload completes.



NOTE

Java binary scan: Upload a ZIP file that contains multiple JAR files or upload a single JAR or WAR file.

Source code scan: Upload a ZIP file or a folder that contains the files you want to scan. The ZIP file or folder can include files with different languages. Contrast Scan automatically detects the languages for the files.

5. [Monitor \(page 669\)](#) the scan progress.

Cancel a scan

You can cancel a scan that is in progress.

After you cancel a scan, its status changes to Cancelled in Scan history.

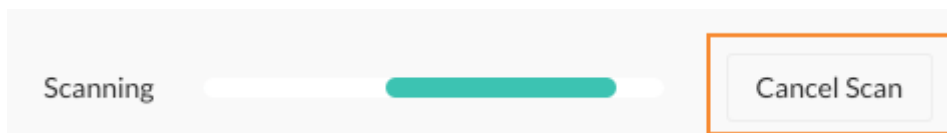
Before you begin

Find a scan project that contains a scan that's in progress.

Steps

To cancel a scan:

1. Select **Scans** in the header.
2. Select a scan project that has a scan in progress.
3. In the activity bar, select **Cancel scan**.



The scan stops immediately.

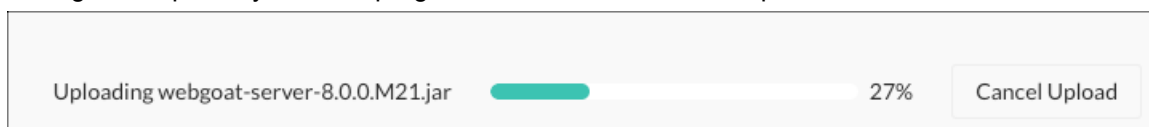
Monitor scans

After you start a scan, you can monitor the progress of the file upload and the scan from any tab under Scans.

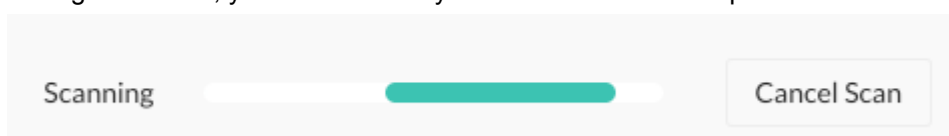
You can view a history of the scans in a selected project under Scan history in the Overview tab.

Steps




1. Select **Scans** in the header.
The Scans page shows these scan details:
 - The grade score for each scan project.
 - The names of the scan projects.
 - The number and status of open vulnerabilities.
 - The time the last scan completed.
2. Select a scan project.
3. [Start a scan \(page 668\)](#).
4. While uploading a file or running a scan, you can monitor its progress at the top of a Scans page.
 - During a file upload, you see a progress bar similar to this example:



- During a file scan, you see an activity bar similar to this example:



5. To view a history of the scans in a selected project, under Scans, select the **Overview** tab and view the details under Scan history.

SCAN HISTORY				
All (3)				
Vulnerabilities	Label	Scan Date	Language	Coverage
 34 7	2022-03-10 20:36:32 1 hour ago	3/10/2022 3:36:32 ...	Java	View
 34 7	2022-03-10 20:34:33 1 hour ago	3/10/2022 3:34:33 ...	Java	View
 34 7	2022-03-10 20:29:43 1 hour ago	3/10/2022 3:29:43 ...	Java	View
<div> <div>1</div> </div>				Results per page 10

- To see additional details about a scan, select a scan label or, under the Coverage. column, select **View**.

SAST reports

Contrast Scan provides these reports in PDF format:


- SAST Attestation report: (page 670)** This report provides evidence of vulnerability remediation based on the most current scan information.
- SAST General Vulnerability report: (page 671)** This report provides details for the vulnerabilities that Contrast discovers.

Generate a SAST Attestation report

The SAST Attestation report for Contrast Scan provides evidence of vulnerability remediation based on the most current scan information. The report is a PDF file.

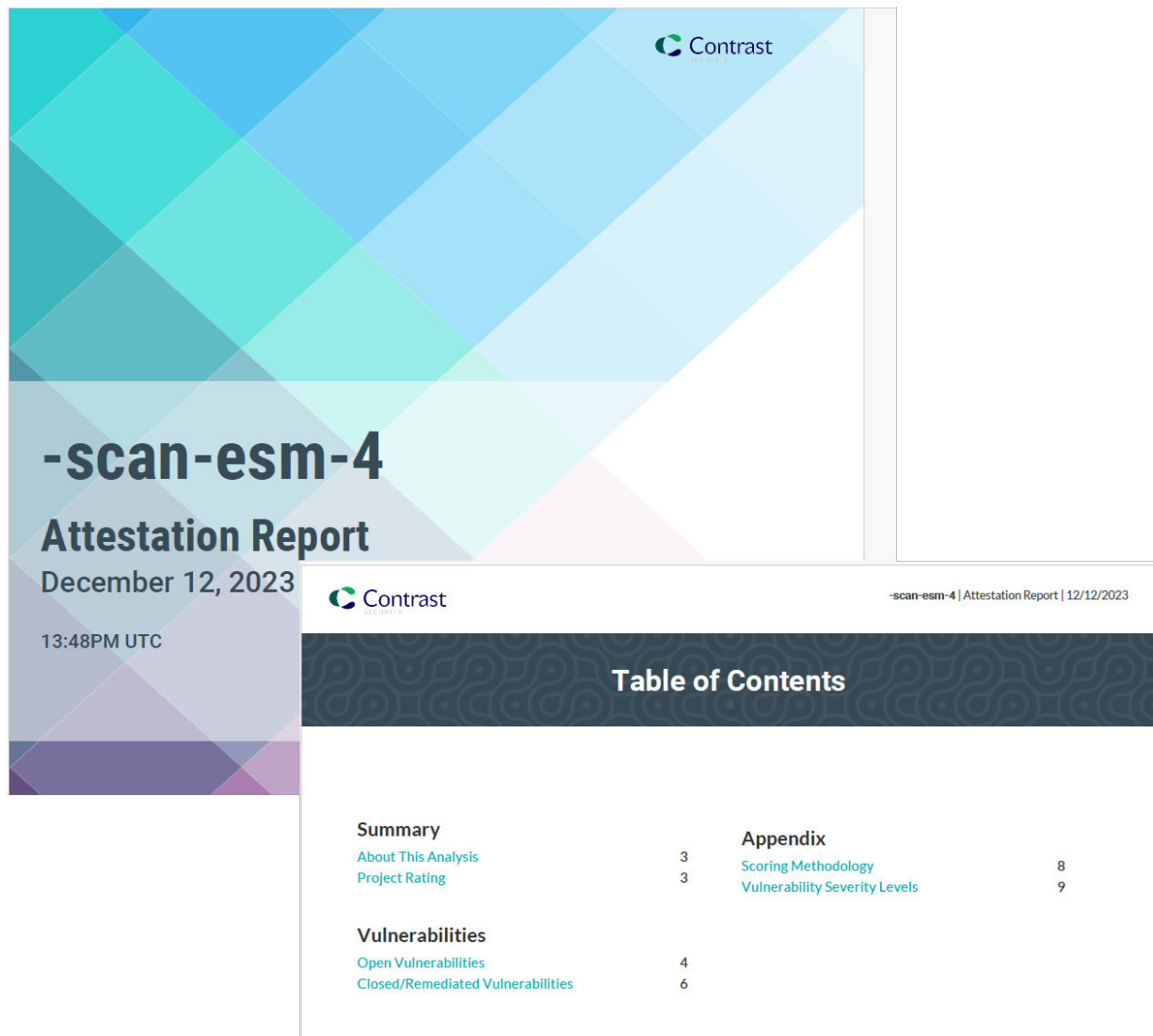
This report includes up to 3,000 entries.

Steps

- Select **Scans** in the header.
- Select a scan project.
- Select the **Reports** icon () located at the top-right of the Scan project page.
You can also generate the report from the Vulnerabilities tab.
- Select **Generate Attestation report**.
- After Contrast generates the report, it prompts you to download it.

SAST attestation report details

For a selected project, the SAST Attestation report includes charts and details about the project score, open vulnerabilities found during scans, and closed or remediated vulnerabilities.



Generate a SAST General Vulnerability report

The SAST General Vulnerability report includes these details:

- **Vulnerability name:** Links to the vulnerability details in the Contrast web interface.
- **Vulnerability type:** For example, cryptography or injection
- **Severity:** The Contrast vulnerability severity: critical, high, medium, low, or note
- **Language:** The language of the code where Contrast found the vulnerability.
- **Vulnerable file name:** Name of file where Contrast discovered the vulnerability..
- **Vulnerable line number:** Line number in the file where Contrast discovered the vulnerability
- **Status:** The Contrast status for the vulnerability: Reported, Confirmed, Suspicious, Not a problem, Remediated, Remediated - Auto-Verified, or Fixed.
- **Code snippet:** A portion of the code where Contrast Scan found the vulnerability.
- **CWE:** (if available): A link to the relevant Common Weakness Enumeration (CWE).
- **OWASP:** A link to the OWASP page for the vulnerability.

This report can include up to 3,000 vulnerabilities.

Steps

1. Select **Scans** in the header.
2. Select a scan project.
3. Select the **Reports** icon (📄) located at the top-right of the Scan project page.
4. Select **Generate SAST General Vulnerability Report**.
5. After Contrast generates the report, it prompts you to download it.

Contrast Scan local engine

The Contrast Scan local engine lets you scan your application using a Java JAR file instead of the Contrast CLI or the Contrast web interface. When a scan completes successfully, the Scan local engine uploads the results to the Contrast platform where you can view them. The uploaded files include:

- Scan results in Static Analysis Results Format (SARIF) in a `JSON` file.
- Output from the scan in a `LOG` file.

This method of scanning is useful if you want to scan files locally without uploading them to the Contrast platform.

Supported platforms

- The Scan local engine is supported for Linux systems.
- **Multi-language Scan local engine:** Oracle Java 17.
- **Binary Java scan engine:** Oracle Java 17.

It's possible to use Oracle Java 11, however, that version is considered to be End-of-Life.

Proxy server settings for local scans

For security purposes, you might want to use a proxy server for communication between the local scan engine and the Contrast platform. Use the following environment variables to enable a proxy server when you [run a local scan \(page 877\)](#):

Variable	Description
CONTRAST__API__PROXY__ENABLE	Enables proxy settings.
CONTRAST__API__PROXY__URL	Required. The URL for the proxy server (for example, <code>http://host:port</code>)
CONTRAST__API__PROXY__TYPE	Required. The proxy server type (for example, BASIC)
CONTRAST__API__PROXY__USERNAME	Optional. Username for the proxy server
CONTRAST__API__PROXY__PASSWORD	Optional. Password for the proxy server

Package preparation for local scans

Consider these best practices when you prepare to run a local scan:

- **JAR or WAR files:** Specify the binary file to be scanned.
- **Source code scanning:** Place the source code you want to scan in a folder and **not** in a ZIP file. There is no limit to the size of this folder for the local scanner. However some large source code repos may require more memory or a longer time to execute. Use the [memory and timeout options \(page 880\)](#) to manage these situations.
- **Multi-JAR or WAR scanning:** Specify a ZIP file that contains multiple JAR or WAR files. Place these files in the root of the ZIP file.

There is no limit to the size of the files in the ZIP file.

Use a consistent file structure

Using a consistent file structure for each scan is crucial to preventing duplicate vulnerability findings. After an initial scan, if you need to change the file structure for the files you are scanning, create a new scan project for those files instead of using an existing one. If you are planning to scan multiple branches without using the [Contrast Scan Analyze](#) GitHub action, create a separate scan project for your personal branch.

If you use an existing scan project, change the file structure, and then run a scan, Contrast sets the original vulnerability status to **Remediated** and reports a new, duplicate vulnerability. The new finding is linked to the same file and line number but shows the new path.

Consistent file structure affects files in ZIP files and files **not** in a ZIP file.

Example 1: Changing ZIP file name

In this example, we rename a ZIP file but maintain the file structure in the ZIP file. This change doesn't result in Contrast reporting duplicate vulnerabilities:

```
scan.zip
|
|-- source_files

changed to

contrastscan.zip
|
|-- source_files
```

In this case, the ZIP file name isn't part of the scan path, so changing it has no effect on the scan findings.

Example 2: Changing file structure in a ZIP file

In this example, we rename the ZIP file and also change the file structure by adding a new directory. This change causes Contrast to report duplicate vulnerabilities.

```
scan.zip
|
|-- source_files

changed to

contrastscan.zip
|
|-- contrastscan
|   |
|   |-- source files
```

In this case, changing the file structure within the ZIP file causes duplicate findings. To avoid this issue, create a new scan project and use it for future scans.

Example 3: Changing file structure in directories

In this example, we change the file structure by adding a new directory. This change causes Contrast to report duplicate vulnerabilities.

```
scan
|
|-- source_files

changed to

contrastscan
|
|-- contrastscan
|
|-- source files
```

In this case, changing the directory name and the file structure within the directory causes duplicate findings. To avoid this issue, create a new scan project and use it for future scans.

Scan process

To use the Scan local engine:

1. [Download the Scan local engine application \(page 875\)](#) or contact [Contrast Support](#) to get the latest local engine application.
2. Decide if you want to use a proxy server for uploading results.
3. [Run the scan on a local system \(page 877\)](#).
4. [View results \(page 886\)](#) in the Contrast web interface.

Create custom Scan rule exclusions

Create custom Scan rule exclusions when you are confident that a specific rule is generating multiple false positives and you are frequently setting the status for these findings to **Not a Problem**.

Before you begin

- Before you exclude rules, run a baseline scan to determine which rules are creating false positives
- Excluding rules affects the findings for the scan project.

Location of Scan rules

The [Contrast Scan rules \(page 675\)](#) section links to tables where you can find the rules for each language. Alternatively, during a scan, you can find the rules in `target/engines/sast-engine4/rulesets` under the source directory that you're scanning with the Scan local engine.



IMPORTANT

The `sast-engines4` folder and its subfolders are available temporarily when a scan is in progress.

For easy access, make a copy of the `rulesets` folder. This folder contains security rules files for each language (`qaking_{lang}_security.xml`) that the Scan local engine supports. To find the name of a rule in a file, search for `<rule name " "`.

Steps

1. Create a text file called `contrastsec.checks.config` and place it at the root of the project you are going to scan.

The format of the file is:

```
[rule-Engine-rule-ID]
active=false
```

2. To exclude multiple rules, repeat the block of lines with an empty row between each block. For example, to exclude the **Detect and handle input/output errors** and **Don't use cast** rules for CPP, the file looks like this:

```
[OPT.CPP.CERTC.FIO33]
active=false

[OPT.CPP.DontUseCast]
active=false
```

Effects of Scan rule exclusions

When you disable rules using the `contrastsec.checks.config` file, the status for the findings corresponding to the excluded rule change to **Remediated**.

If you reenables the rule using the `contrastsec.checks.config` file or remove the file, the status for the findings corresponding to the newly enabled rule changes to **Reopened**.

Contrast Scan rules

These tables include the supported Contrast Scan rules: test

ABAP (page 675)	Go (page 745)	NATURAL (page 807)	Scala (page 844)
ActionScript (page 687)	HTML (page 748)	Objective-C (page 808)	SQL (page 846)
ASP (page 691)	Informix (page 751)	Oracle Forms (page 816)	SQLScript (page 849)
ASP.NET (page 689)	Java (page 753)	PHP (page 818)	Swift (page 850)
C (page 734)	JavaScript (page 789)	PL/SQL (page 827)	Transact-SQL (page 854)
C# (page 692)	JCL (page 800)	PowerScript (page 832)	VB.NET (page 857)
COBOL (page 707)	JSP (page 801)	Python (page 834)	Visual Basic 6 (page 871)
CPP (page 721)	Kotlin (page 803)	RPG4 (page 840)	XML (page 873)

ABAP Scan rules

Contrast Scan supports these rules for ABAP.

Severity	Contrast rule	Engine rule ID	Description
Medium	Access B Din Loop	OPT.ABAP.AMR.AccessBDInLoop	AccessBDInLoop: Avoid massive database operations inside a loop
High	Alter Layout Dinamically	OPT.ABAP.AWD.AlterLayoutDinamically	AlterLayoutDinamically: WebDynpro Layout should be modified only in wdDoModifyView method
High	Assign I D Element	OPT.ABAP.AWD.AssignIDElement	AssignIDElement: Attributes for Web Dynpro elements must be unique
Info	Authority Checks	OPT.ABAP.SEC.AuthorityChecks	AuthorityChecks: Authority checks (informative)
Medium	Avoid Batch Input	OPT.ABAP.AMR.AvoidBatchInput	AvoidBatchInput: Do not call transactions using batch input

Severity	Contrast rule	Engine rule ID	Description
Medium	Avoid Call No Def Module	OPT.ABAP.APBR.AvoidCallNoDefModule	AvoidCallNoDefModule: Avoid call modules that have not been declared
High	Avoid Client Specified	OPT.ABAP.ASR.AvoidClientSpecified	AvoidClientSpecified: Avoid CLIENT SPECIFIED option
Low	Avoid Commented Out Code	OPT.ABAP.MAINT.AvoidCommentedOutCode	AvoidCommentedOutCode: Avoid commented out code blocks
Medium	Avoid Complex Context	OPT.ABAP.AWD.AvoidComplexContext	AvoidComplexContext: Too deeply nested nodes in Web Dynpro context
Medium	Avoid Controller With Code	OPT.ABAP.AWD.AvoidControllerWithCode	AvoidControllerWithCode: Too much code in Web Dynpro view
High	Avoid Database Hints	OPT.ABAP.PORTABILITY.AvoidDatabaseHints	AvoidDatabaseHints: Avoid %_HINTS in SELECT
Medium	Avoid Declare Vars In Mod	OPT.ABAP.AGR.AvoidDeclareVarsInMod	AvoidDeclareVarsInMod: Avoid declarations inside a dialog module
Medium	Avoid Duplicate Events	OPT.ABAP.AGR.AvoidDuplicateEvents	AvoidDuplicateEvents: Avoid duplicate declarations for same event block
Info	Avoid Duplicate Includes In Programs	OPT.ABAP.APFR.AvoidDuplicateIncludesInPrograms	AvoidDuplicateIncludesInPrograms: Avoid same INCLUDE in different programs
Low	Avoid Elementil U Tree	OPT.ABAP.AWD.AvoidElementilUTree	AvoidElementilUTree: In Web Dynpro, do not use the Tree UI element
Low	Avoid Empty Blocks In Loop Or If	OPT.ABAP.MAINT.AvoidEmptyBlocksInLoopOrIf	AvoidEmptyBlocksInLoopOrIf: Avoid using loops and conditional statements with empty blocks
Medium	Avoid Empty Catch Blocks	OPT.ABAP.APBR.AvoidEmptyCatchBlocks	AvoidEmptyCatchBlocks: Avoid use empty CATCH blocks
Low	Avoid Empty Subroutine Or Function	OPT.ABAP.MAINT.AvoidEmptySubroutineOrFunction	AvoidEmptySubroutineOrFunction: Avoid using functions or subroutines with empty blocks
Medium	Avoid Empty When Others	OPT.ABAP.APBR.AvoidEmptyWhenOthers	AvoidEmptyWhenOthers: If used, WHEN OTHERS clause should not be empty
Low	Avoid Form Param Without Type	OPT.ABAP.AGR.AvoidFormParamWithoutType	AvoidFormParamWithoutType: Avoid subroutines with untyped or too generic type parameters
Critical	Avoid Free Memory	OPT.ABAP.AGR.AvoidFreeMemory	AvoidFreeMemory: Avoid using FREE MEMORY without explicit data cluster
Low	Avoid From Dynamic	OPT.ABAP.ASR.AvoidFromDynamic	AvoidFromDynamic: Avoid subqueries in FROM clauses
Medium	Avoid Literal Wit Add	OPT.ABAP.AGR.AvoidLiteralWitAdd	AvoidLiteralWitAdd: In the sentece Add is better use variables instead of literal
Low	Avoid Logic DB	OPT.ABAP.ASR.AvoidLogicDB	AvoidLogicDB: Do not to use logical databases
High	Avoid Macro	OPT.ABAP.AGR.AvoidMacro	AvoidMacro: Do not define custom macros
Low	Avoid Modify Context	OPT.ABAP.AWD.AvoidModifyContext	AvoidModifyContext: Avoid programmatic modification of a Web Dynpro context

Severity	Contrast rule	Engine rule ID	Description
Low	Avoid Modify Elements Directly	OPT.ABAP.AWD.AvoidModifyElementsDirectly	AvoidModifyElementsDirectly: Do not modify programmatically the layout for Web Dynpro context nodes
Low	Avoid Module Messed Up	OPT.ABAP.APBR.AvoidModuleMessedUp	AvoidModuleMessedUp: Avoid call a module from a program
Low	Avoid Multiple Dynamic Attributes	OPT.ABAP.AWD.AvoidMultipleDynamicAttributes	AvoidMultipleDynamicAttributes: Avoid assigning multiple attributes dynamically in a WebDynpro element
Low	Avoid Nested Layout	OPT.ABAP.AWD.AvoidNestedLayout	AvoidNestedLayout: To avoid to nest layouts
Low	Avoid Non Transp Table	OPT.ABAP.AGR.AvoidNonTranspTable	AvoidNonTranspTable: Do not create tables that are not transparent
Medium	Avoid Non Used Declared Module	OPT.ABAP.AMR.AvoidNonUsedDeclaredModule	AvoidNonUsedDeclaredModule: Avoid declaring dialog modules that are not used
Info	Avoid No Standard Page Heading	OPT.ABAP.ADR.AvoidNoStandardPageHeading	AvoidNoStandardPageHeading: The standard SAP report header should not be disabled (NO STANDARD PAGE HEADING)
Medium	Avoid No Use Fields	OPT.ABAP.AGR.AvoidNoUseFields	AvoidNoUseFields: Avoid declare fields on internal tables that are not used
High	Avoid Percentage Var	OPT.ABAP.AMR.AvoidPercentage_Var	AvoidPercentage_Var: Avoid identifier names starting with %_
Medium	Avoid Pool And Include	OPT.ABAP.AWD.AvoidPoolAndInclude	AvoidPoolAndInclude: Do not use TYPE-POOL and INCLUDE TYPE/STRUCTURE
High	Avoid Read S A P Tables	OPT.ABAP.ASR.AvoidReadSAPTables	AvoidReadSAPTables: READ TABLE on database table
Medium	Avoid S A P Queries	OPT.ABAP.ASR.AvoidSAPQueries	AvoidSAPQueries: Do not call SAP Query
Medium	Avoid Sap Script	OPT.ABAP.AGR.AvoidSapScript	AvoidSapScript: Avoid SAP Script
High	Avoid SQL Exist Subqueries	OPT.ABAP.EFFICIENCY.AvoidSqlExistSubqueries	AvoidSqlExistSubqueries: Avoid using the EXISTS clause
Medium	Avoid Stop Messed Up	OPT.ABAP.APBR.AvoidStopMessedUp	AvoidStopMessedUp: Avoid STOP statement out of the event blocks START-OF-SELECTION, GET or AT SELECTION-SCREEN
High	Avoid Stretched Vertically	OPT.ABAP.AWD.AvoidStretchedVertically	AvoidStretchedVertically: Do not set StretchedVertically in MatrixLayout
Medium	Avoid Sub Select Queries	OPT.ABAP.APFR.AvoidSubSelectQueries	AvoidSubSelectQueries: Avoid subqueries in WHERE
High	Avoid Trace On	OPT.ABAP.APFR.AvoidTraceOn	AvoidTraceOn: Avoid use sentences that activate the trace generator
Critical	Avoid Up To Rows With For All Entries	OPT.ABAP.EFFICIENCY.AvoidUpToRowsWithForAllEntries	AvoidUpToRowsWithForAllEntries: Avoid using UP TO ROWS clause in combination with FOR ALL ENTRIES in SELECT statements
Info	Avoid Vble Message	OPT.ABAP.ADR.AvoidVbleMessage	AvoidVbleMessage: Do not use variables as parameters for messages
Low	Avoid Where Dynamic	OPT.ABAP.ASR.AvoidWhereDynamic	AvoidWhereDynamic: Avoid subqueries in WHERE

Severity	Contrast rule	Engine rule ID	Description
Info	Avoid Write To	OPT.ABAP.AGR.AvoidWriteTo	AvoidWriteTo: Replace WRITE TO sentence with MOVE TO
Critical	Backdoors	OPT.ABAP.SEC.Backdoors	Backdoors: Avoid development/test backdoors in production code
High	Bad Authorization Check	OPT.ABAP.SEC.BadAuthorizationCheck	BadAuthorizationCheck: Improper implementation of authorization check
High	Call Editor Call	OPT.ABAP.AGR.CallEditorCall	CallEditorCall: Avoid EDITOR-CALL
Medium	Call F M In Group	OPT.ABAP.AGR.CallFMInGroup	CallFMInGroup: Unused function
Info	Calls2 Critical Functions	OPT.ABAP.SEC.Calls2CriticalFunctions	Calls2CriticalFunctions: Calls to Critical ABAP functions
Critical	Call Sys Function	OPT.ABAP.AGR.CallSysFunction	CallSysFunction: Do not call system / kernel functions from ABAP application code
Low	Call S Y S U B R C	OPT.ABAP.APBR.CallSYSUBRC	CallSYSUBRC: Check the value of SY-SUBRC after certain operations
High	Call Tx	OPT.ABAP.AGR.CallTx	CallTx: Avoid called transactions corresponding to a certain module
Medium	Case No Repeat When	OPT.ABAP.AGR.CaseNoRepeatWhen	CaseNoRepeatWhen: Avoid repeat WHEN conditions in a CASE sentence
Low	Case Should Have At Least3 When	OPT.ABAP.MAINT.CaseShouldHaveAtLeast3When	CaseShouldHaveAtLeast3When: A CASE statement should have at least 3 WHEN clauses
Low	Check Atr No Exist Dynpro	OPT.ABAP.APBR.CheckAtrNoExistDynpro	CheckAtrNoExistDynpro: Avoid checking attributes of a non-existing dynpro
Medium	Check Auth In All Programs	OPT.ABAP.SEC.CheckAuthInAllPrograms	CheckAuthInAllPrograms: Any report must perform an authority check
High	Check Authority	OPT.ABAP.APBR.CheckAuthority	CheckAuthority: Invalid authorization fields in AUTHORITY-CHECK
High	Check Dlg Modules	OPT.ABAP.APBR.CheckDlgModules	CheckDlgModules: Avoid work with non-existing dynpros
Medium	Check F Don't use masterpage filesaram	OPT.ABAP.APBR.CheckFMPParam	CheckFMPParam: Avoid use an incorrect category for a function module parameters
High	Check Fn Module	OPT.ABAP.APBR.CheckFnModule	CheckFnModule: Avoid working with non-existing functions
Low	Check Includes	OPT.ABAP.APBR.CheckIncludes	CheckIncludes: Avoid call files from include sentece that are not type I
High	Check Load Table	OPT.ABAP.APBR.CheckLoadTable	CheckLoadTable: Load only existing tables in SAP system
High	Check Messages	OPT.ABAP.APBR.CheckMessages	CheckMessages: Use only existing messages in the table T100
Info	Check Status P F	OPT.ABAP.AGR.CheckStatusPF	CheckStatusPF: Check if the program program personalizes the STATUS PF
Medium	Check Submit With Param	OPT.ABAP.APBR.CheckSubmitWithParam	CheckSubmitWithParam: Avoid undeclared selection criteria in SUBMIT
Info	Check Titlebar	OPT.ABAP.AGR.CheckTitlebar	CheckTitlebar: Check if the program program personalizes the TITLEBAR
High	Check Tx	OPT.ABAP.APBR.CheckTx	CheckTx: Only use existing transactions

Severity	Contrast rule	Engine rule ID	Description
High	Close All Open Resources	OPT.ABAP.APFR.CloseAllOpenResources	CloseAllOpenResources: Every open resource (cursor or dataset) should be closed
Low	Cmd Table Out Loop	OPT.ABAP.ASR.CmdTableOutLoop	CmdTableOutLoop: Avoid to use commands of table with implicit use of index out of the LOOP
Critical	Command Injection	OPT.ABAP.SEC.CommandInjection	CommandInjection: Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
Low	Comments Before Classes	OPT.ABAP.MAINT.CommentsBeforeClasses	CommentsBeforeClasses: Check if there are comment lines before a class
High	Comments Before Programs Or Reports	OPT.ABAP.ADR.CommentsBeforeProgramsOrReports	CommentsBeforeProgramsOrReports: Check if there are comment lines before programs and reports
Low	Comments Before Subroutines	OPT.ABAP.ADR.CommentsBeforeSubroutines	CommentsBeforeSubroutines: Check if there are comment lines before functions, method, forms or macro
Medium	Compatible Form Params	OPT.ABAP.APBR.CompatibleFormParams	CompatibleFormParams: With PERFORM, arguments should match subroutine formal parameters
Medium	Complex Layout Use Matrix Layout	OPT.ABAP.AWD.ComplexLayoutUseMatrixLayout	ComplexLayoutUseMatrixLayout: Avoid too many elements in Web Dynpro views
High	Control Fields Client Tables	OPT.ABAP.ASR.ControlFieldsClientTables	ControlFieldsClientTables: Include audit fields in custom tables
Low	Correct Naming Meth	OPT.ABAP.ADR.CorrectNamingMeth	CorrectNamingMeth: Methods must follow a standard naming convention
High	Correspond Raise Excep	OPT.ABAP.APBR.CorrespondRaiseExcep	CorrespondRaiseExcep: EXCEPTIONS and RAISE must match
Critical	Cross Client Database Access	OPT.ABAP.SEC.CrossClientDatabaseAccess	CrossClientDatabaseAccess: Do not bypass SAP client separation mechanism
Critical	Cross Site Scripting	OPT.ABAP.SEC.CrossSiteScripting	CrossSiteScripting: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
High	Cx Root Caught	OPT.ABAP.RELIABILITY.CxRootCaught	CxRootCaught: Do not catch CX_ROOT
High	Cyclomatic Complexity	OPT.ABAP.AMTR.CyclomaticComplexity	CyclomaticComplexity: Avoid functions/forms/methods...." with high cyclomatic complexity
Info	Dangerous File Download	OPT.ABAP.SEC.DangerousFileDownload	DangerousFileDownload: Dangerous file download
Info	Dangerous File Upload	OPT.ABAP.SEC.DangerousFileUpload	DangerousFileUpload: Dangerous file upload
Low	Data Definition At The Beginning	OPT.ABAP.AMR.DataDefinitionAtTheBeginning	DataDefinitionAtTheBeginning: Do not insert declarations after the first executable line
Medium	Deeply Nested Statements	OPT.ABAP.MAINT.DeeplyNestedStatements	DeeplyNestedStatements: Avoid too deeply nested statements

Severity	Contrast rule	Engine rule ID	Description
High	Delete From Table Without Where	OPT.ABAP.RELIABILITY.DeleteFromTableWithoutWhere	DeleteFromTableWithoutWhere: DELETE FROM statement must have WHERE condition
High	Deprecated Asynchronous R F C	OPT.ABAP.PORTABILITY.DeprecatedAsynchronousRFC	DeprecatedAsynchronousRFC: Use Background RFC instead of older Transactional or Queue RFC
High	Direct Recursive Call	OPT.ABAP.RELIABILITY.DirectRecursiveCall	DirectRecursiveCall: Avoid recursive calls
High	Direct Update	OPT.ABAP.SEC.DirectUpdate	DirectUpdate: SQL Bad Practices - Direct Update
Critical	Dynamic Code	OPT.ABAP.SEC.DynamicCode	DynamicCode: Avoid Dynamic Code constructs
High	Dynamic Constructs	OPT.ABAP.SEC.DynamicConstructs	DynamicConstructs: Avoid dynamic constructs controlled by external input
Medium	Dynpro Non Exist Atr	OPT.ABAP.APBR.DynproNonExistAtr	DynproNonExistAtr: Avoid references to non-existing attributes for a dynpro screen element
Low	Elseif With Else	OPT.ABAP.RELIABILITY.ElseifWithElse	ElseifWithElse: IF ... ELSEIF statements must be terminated with ELSE
Critical	Empty Select Endselect	OPT.ABAP.EFFICIENCY.EmptySelectEndselect	EmptySelectEndselect: Avoid empty SELECT-ENDSELECT statements
Critical	Enqueue Instead Of Select Single For Update	OPT.ABAP.ASR.EnqueueInsteadOfSelectSingleForUpdate	EnqueueInsteadOfSelectSingleForUpdate: Use SAP locking mechanism instead of SELECT SINGLE FOR UPDATE
Low	Equal Number Param	OPT.ABAP.APBR.EqualNumberParam	EqualNumberParam: Avoid call a FORM with diferent number of parameters as declared
Low	Excess Of Parameters	OPT.ABAP.AMTR.ExcessOfParameters	ExcessOfParameters: Avoid functions/forms/methods with an excess of parameters
Low	Excess Of Responsibility	OPT.ABAP.AMTR.ExcessOfResponsibility	ExcessOfResponsibility: Avoid programs with excess of responsibility
Low	Excess Of Return	OPT.ABAP.AMTR.ExcessOfReturn	ExcessOfReturn: Avoid an excess of RETURN statements
Low	Exist Prog Form Decl	OPT.ABAP.APBR.ExistProgFormDecl	ExistProgFormDecl: Avoid declare FORMS in programs that do not exists in the analyzed system
Medium	Exists Form	OPT.ABAP.AGR.ExistsForm	ExistsForm: Avoid call non-declared FORMS
Medium	Extract Code Into Subroutines	OPT.ABAP.AMR.ExtractCodeIntoSubroutines	ExtractCodeIntoSubroutines: Organize big code sections in smaller ones
Low	Fan In	OPT.ABAP.AMTR.FanIn	FanIn: Limit the number of calls to each routine in programs, reports and classes
Medium	Fan Out	OPT.ABAP.AMTR.FanOut	FanOut: Limit number of calls (fan-out) from each processing block
Info	Fields Curr Quan	OPT.ABAP.AGR.FieldsCurrQuan	FieldsCurrQuan: CURR and QUAN fields must have asociated unit fields
Critical	First Stmt Return Select Endselect	OPT.ABAP.EFFICIENCY.FirstStmtReturnSelectEndselect	FirstStmtReturnSelectEndselect: Avoid using a RETURN statement as the first statement into a SELECT-ENDSELECT block
High	Form Corresponds Perform	OPT.ABAP.AGR.FormCorrespondsPerform	FormCorrespondsPerform: Avoid declaring unused subroutines

Severity	Contrast rule	Engine rule ID	Description
Low	Hardcoded Text In Message	OPT.ABAP.MAINT.HardcodedTextInMessage	HardcodedTextInMessage: Avoid hardcoding text in message
High	Hardcoded Client Check	OPT.ABAP.SEC.HardcodedClientCheck	HardcodedClientCheck: Hardcoded SAP client check (sy-mandt)
Low	Hardcoded Date Check	OPT.ABAP.SEC.HardcodedDateCheck	HardcodedDateCheck: Avoid hardcoding into the code current server date checks (sy-datum)
Medium	Hardcoded Host Check	OPT.ABAP.MAINT.HardcodedHostCheck	HardcodedHostCheck: Avoid hardcoding sy-host checks
Low	Hardcoded OS Check	OPT.ABAP.MAINT.HardcodedOSCheck	HardcodedOSCheck: Avoid hardcoding sy-opsys checks
Medium	Hardcoded Rfc Destination	OPT.ABAP.RELIABILITY.HardcodedRfcDestination	HardcodedRfcDestination: Avoid hardcoding the RFC destination parameter
Low	Hardcoded Sensitive Data	OPT.ABAP.SEC.HardcodedSensitiveData	HardcodedSensitiveData: Avoid hardcoding sensitive information
High	Hardcoded System Id Check	OPT.ABAP.SEC.HardcodedSystemIdCheck	HardcodedSystemIdCheck: Hardcoded System ID check (sy-sysid)
High	Hardcoded Username Check	OPT.ABAP.SEC.HardcodedUsernameCheck	HardcodedUsernameCheck: Hard-coded user name in check (potential backdoor)
Critical	Http Header Manipulation	OPT.ABAP.SEC.HttpHeaderManipulation	HttpHeaderManipulation: Unvalidated data in HTTP response header
Medium	Implement Layout	OPT.ABAP.AWD.ImplementLayout	ImplementLayout: All Web Dynpro views must have a layout
Low	Import Export Dynpro	OPT.ABAP.AMR.ImportExportDynpro	ImportExportDynpro: Avoid using import/export dynpro
Critical	Import Export Nametab	OPT.ABAP.AGR.ImportExportNametab	ImportExportNametab: Avoid IMPORT / EXPORT NAMETAB
High	Insecure Randomness	OPT.ABAP.SEC.InsecureRandomness	InsecureRandomness: Standard pseudo-random number generators cannot withstand cryptographic attacks
High	Join Instead Of Select In Loop	OPT.ABAP.EFFICIENCY.JoinInsteadOfSelectInLoop	JoinInsteadOfSelectInLoop: Use join instead of select + loop + nested select
High	Key Select Option	OPT.ABAP.AGR.KeySelectOption	KeySelectOption: SELECT-OPTIONS must always include key or indexed fields
Info	Keywords For User Identifiers	OPT.ABAP.MAINT.KeywordsForUserIdentifiers	KeywordsForUserIdentifiers: ABAP keywords should not be used as identifiers
Info	Limited Number Of Includes	OPT.ABAP.APFR.LimitedNumberOfIncludes	LimitedNumberOfIncludes: Too many included source files
High	Limited Number Of Tables In Queries	OPT.ABAP.ASR.LimitedNumberOfTablesInQueries	LimitedNumberOfTablesInQueries: Try to limit the number of tables used in a SELECT query
Medium	Limit Indexes	OPT.ABAP.ASR.LimitIndexes	LimitIndexes: Do not to create too many indexes in a table
Medium	Logic Depending On Text Symbols	OPT.ABAP.RELIABILITY.LogicDependingOnTextSymbols	LogicDependingOnTextSymbols: Logic depending on text symbols

Severity	Contrast rule	Engine rule ID	Description
Info	Loop At Into	OPT.ABAP.EFFICIENCY.LoopAtInto	LoopAtInto: Avoid LOOP AT itab INTO
Low	Loop At Where Inside Loop	OPT.ABAP.APFR.LoopAtWhereInsideLoop	LoopAtWhereInsideLoop: Avoid using LOOP AT with WHERE inside another LOOP AT
Info	Loop Where Instead Of Loop Check	OPT.ABAP.APFR.LoopWhereInsteadOfLoopCheck	LoopWhereInsteadOfLoopCheck: Use LOOP + WHERE instead of LOOP + CHECK
High	Mark Buffering If Necessary	OPT.ABAP.ASR.MarkBufferingIfNecessary	MarkBufferingIfNecessary: Activate table buffering when needed
Medium	Match Layout With View Controller	OPT.ABAP.AWD.MatchLayoutWithViewController	MatchLayoutWithViewController: Without associated layout, view controller code is stored in wdDoModifyView method
Low	Maximum Joins Per Query	OPT.ABAP.APFR.MaximumJoinsPerQuery	MaximumJoinsPerQuery: Restrict the number of tables that can be used in a SELECT
Low	Max Long Line Size	OPT.ABAP.APTR.MaxLongLineSize	MaxLongLineSize: Avoid width parameter LINE-SIZE greater than 255
Medium	Max Number DB Op	OPT.ABAP.APFR.MaxNumberDBOp	MaxNumberDBOp: Too many DB operations in a report or procedure
Low	Max One Append Struct	OPT.ABAP.ASR.MaxOneAppendStruct	MaxOneAppendStruct: Table or standard structure with multiple append structures
Low	Message Language	OPT.ABAP.ADR.MessageLanguage	MessageLanguage: The messages must be translated into the necessary languages
Low	Message Param	OPT.ABAP.APBR.MessageParam	MessageParam: Check that messages have the correct number of parameters
High	Modified Input Parameter	OPT.ABAP.RELIABILITY.ModifiedInputParameter	ModifiedInputParameter: Modification of input parameter passed by reference
Info	Move Instead Of Move Corresponding	OPT.ABAP.AGR.MoveInsteadOfMoveCorresponding	MoveInsteadOfMoveCorresponding: Use MOVE instead of MOVE-CORRESPONDING
High	Naming Components	OPT.ABAP.AWD.NamingComponents	NamingComponents: To control the nomenclature of the components
Low	Naming Conventions	OPT.ABAP.ADR.NamingConventions	NamingConventions: Standard naming conventions should be followed
Low	Nested Case Statement	OPT.ABAP.AMR.NestedCaseStatement	NestedCaseStatement: Avoid CASE sentences with many nested levels
Low	Nested If Statement	OPT.ABAP.AMR.NestedIfStatement	NestedIfStatement: Try to limit nested IF
Medium	Nested Loops	OPT.ABAP.AMR.NestedLoops	NestedLoops: Avoid using nested loops
Medium	Nested Only Data Def	OPT.ABAP.AGR.NestedOnlyDataDef	NestedOnlyDataDef: Avoid declaring a redundant nested structure
Critical	Nested Select Statement	OPT.ABAP.EFFICIENCY.NestedSelectStatement	NestedSelectStatement: Avoid nested SELECT statements
Critical	No Absolute Paths	OPT.ABAP.APTR.NoAbsolutePaths	NoAbsolutePaths: There should not be an absolute path
High	No Access Model From Mths	OPT.ABAP.AWD.NoAccessModelFromMths	NoAccessModelFromMths: No direct access to assistance classes from Web Dynpro view controller methods

Severity	Contrast rule	Engine rule ID	Description
Medium	No Append Sorted By	OPT.ABAP.APFR.NoAppendSortedBy	NoAppendSortedBy: Do not use APPEND ... SORTED BY
High	No Assert With Side Effects Condition	OPT.ABAP.RELIABILITY.NoAssertWithSideEffectsCondition	NoAssertWithSideEffectsCondition: Avoid configurable ASSERTs with side-effects
Critical	No Authorization Check Call Transaction	OPT.ABAP.SEC.NoAuthorizationCheckCallTransaction	NoAuthorizationCheckCallTransaction : Authorization check must be done explicitly before CALL TRANSACTION
Critical	No Authorization Check RFC	OPT.ABAP.SEC.NoAuthorizationCheckRFC	NoAuthorizationCheckRFC: Authorization check must be done explicitly in RFC-enabled functions
High	No Authorization Check SQL	OPT.ABAP.SEC.NoAuthorizationCheckSQL	NoAuthorizationCheckSQL: Authorization check must be done explicitly on SQL statements
High	No Authorization Group4 Table	OPT.ABAP.SEC.NoAuthorizationGroup4Table	NoAuthorizationGroup4Table: Table without authorization group
Critical	No Break Point Statements	OPT.ABAP.APBR.NoBreakPointStatements	NoBreakPointStatements: Remove BREAK-POINT statements from production code
Medium	No Check Or Continue Within Select Loops	OPT.ABAP.APFR.NoCheckOrContinueWithinSelectLoops	NoCheckOrContinueWithinSelectLoops: Do not use CHECK, EXIT or CONTINUE statements inside a SELECT loop
High	No Control Break Within Loop At Where	OPT.ABAP.APBR.NoControlBreakWithinLoopAtWhere	NoControlBreakWithinLoopAtWhere: Do not use WHERE/FROM/TO in LOOP AT with control level processing blocks (AT FIRST, AT NEW, AT END, AT LAST)
Low	No Corresponding Fields	OPT.ABAP.APFR.NoCorrespondingFields	NoCorrespondingFields: Do not use CORRESPONDING FIELDS in SELECT *
Medium	No Data Definitions Within Events	OPT.ABAP.AGR.NoDataDefinitionsWithinEvents	NoDataDefinitionsWithinEvents: Avoid declarations inside an event block
Medium	No Data Or Object Creation Inside Loops	OPT.ABAP.APFR.NoDataOrObjectCreationInsideLoops	NoDataOrObjectCreationInsideLoops: Avoid creating objects or declaring data inside a loop
Low	No Dead Data Definitions	OPT.ABAP.AMR.NoDeadDataDefinitions	NoDeadDataDefinitions: In a program, all information defined is in use
Info	No Declared Field	OPT.ABAP.ASR.NoDeclaredField	NoDeclaredField: Avoid referencing undeclared fields in internal tables or structures
High	No Exec SQL Statements	OPT.ABAP.APTR.NoExecSqlStatements	NoExecSqlStatements: There should not be an EXEC SQL statements
Info	No Exist Context	OPT.ABAP.APBR.NoExistContext	NoExistContext: Do not use fields from non declared tables or structures
Medium	No Field As Operator	OPT.ABAP.ADR.NoFieldAsOperator	NoFieldAsOperator: Avoid using operators as field names
Low	No Hyphen In Name	OPT.ABAP.AGR.NoHyphenInName	NoHyphenInName: Not use hyphens in the name of the internal table fields
Medium	No Ids Other Type	OPT.ABAP.ADR.NoIdsOtherType	NoldsOtherType: Field name is identical to a primitive type but it is declared as another type

Severity	Contrast rule	Engine rule ID	Description
Info	No Literals	OPT.ABAP.AMR.NoLiterals	NoLiterals: Do not use literals in the code
Low	No Long Ids	OPT.ABAP.ADR.NoLongIds	NoLongIds: Do not use large identifier names
Info	No Lost Cursor	OPT.ABAP.ASR.NoLostCursor	NoLostCursor: Avoid in a loop SELECT to call to a sentence because of which the cursor gets lost
High	No Over Write Sys Var	OPT.ABAP.AGR.NoOverWriteSysVar	NoOverWriteSysVar: Avoid overwrite system-variables
Medium	No Raise Out Of Function Group	OPT.ABAP.APBR.NoRaiseOutOfFunctionGroup	NoRaiseOutOfFunctionGroup: RAISE exception statements inside improper processing block
Medium	No Select All	OPT.ABAP.ASR.NoSelectAll	NoSelectAll: Avoid SELECT * in SQL queries
High	No Select Inside Loop	OPT.ABAP.APFR.NoSelectInsideLoop	NoSelectInsideLoop: Avoid include SELECT sentences inside a loop
Medium	No Sentence After Exit	OPT.ABAP.AGR.NoSentenceAfterExit	NoSentenceAfterExit: Avoid other sentences after STOP, LEAVE, PROGRAM, EXIT, RETURN, RAISE, REJECT or SUBMIT
High	Not In Subquery	OPT.ABAP.EFFICIENCY.NotInSubquery	NotInSubquery: Avoid NOT IN subquery in SELECT
Critical	No Update Config Tables	OPT.ABAP.ASR.NoUpdateConfigTables	NoUpdateConfigTables: Avoid write operations on sensitive database tables from ABAP code
High	No Use S Y Uname	OPT.ABAP.AGR.NoUseSYUname	NoUseSYUname: Avoid using system variable SY-UNAME inside a condition
High	No Wildcards At The Beginning Of Like Literals	OPT.ABAP.APFR.NoWildcardsAtTheBeginningOfLikeLiterals	NoWildcardsAtTheBeginningOfLikeLiterals: Do not use wildcards (% or _) at the beginning of the literal used in LIKE comparisons in SQL statements
Medium	Number View In W D	OPT.ABAP.AWD.NumberViewInWD	NumberViewInWD: Control the number of views
Medium	Obsolete Code	OPT.ABAP.PORTABILITY.ObsoleteCode	ObsoleteCode: Obsolete code in SAP 7+
Info	One Statement Per Line	OPT.ABAP.AMR.OneStatementPerLine	OneStatementPerLine: Place each sentence in one line
Low	Only Client Fields In Append Str	OPT.ABAP.ASR.OnlyClientFieldsInAppendStr	OnlyClientFieldsInAppendStr: Always use custom fields in Append structures
High	Only One Commit And Rollback	OPT.ABAP.APFR.OnlyOneCommitAndRollback	OnlyOneCommitAndRollback: There should be only one COMMIT and ROLLBACK inside a programme
Medium	Open Close Resources Only Once	OPT.ABAP.APFR.OpenCloseResourcesOnlyOnce	OpenCloseResourcesOnlyOnce: Every resource (cursor or dataset) should be opened and closed only once
Critical	Open Redirect	OPT.ABAP.SEC.OpenRedirect	OpenRedirect: URL Redirection to Untrusted Site ('Open Redirect')
Low	Output File Extension	OPT.ABAP.AGR.OutputFileExtension	OutputFileExtension: Output files must have appropriate extensions
Medium	Output Files With Header	OPT.ABAP.AGR.OutputFilesWithHeader	OutputFilesWithHeader: Output datasets must have a header record
High	Overwrite System Fields	OPT.ABAP.SEC.OverwriteSystemFields	OverwriteSystemFields: Inadequate usage of ABAP System field

Severity	Contrast rule	Engine rule ID	Description
High	Password Management	OPT.ABAP.SEC.PasswordManagement	PasswordManagement: Avoid hard-coded or in-comment credentials (username / password) in code
Critical	Path Manipulation	OPT.ABAP.SEC.PathManipulation	PathManipulation: External Control of File Name or Path
Medium	Path Output File	OPT.ABAP.AMR.PathOutputFile	PathOutputFile: Output files must be stored in file system
Medium	Percentage Of Comment Lines Per File	OPT.ABAP.ADR.PercentageOfCommentLinesPerFile	PercentageOfCommentLinesPerFile: Check the global amount of comment lines per file
Medium	Percentage Of Comment Lines Per Method	OPT.ABAP.MAINT.PercentageOfCommentLinesPerMethod	PercentageOfCommentLinesPerMethod: Check the global amount of comment lines per method
Medium	Recommendable Dynpro Size	OPT.ABAP.AWD.RecommendableDynproSize	RecommendableDynproSize: Use correct sizes in Web Dynpro fields
Medium	Recommend ALV with Report	OPT.ABAP.AGR.RecommendALVwithReport	RecommendALVwithReport: Use ALV (Abap List Viewer) instead of classic list generation in reports
Low	Recommend Case When Others	OPT.ABAP.AGR.RecommendCaseWhenOthers	RecommendCaseWhenOthers: WHEN OTHERS clause should appear in every CASE statement
Info	Recommend Start Of Selection	OPT.ABAP.AGR.RecommendStartOfSelection	RecommendStartOfSelection: START-OF-SELECTION event handler should appear in every ABAP report
High	Recommend Where With Indexes	OPT.ABAP.ASR.RecommendWhereWithIndexes	RecommendWhereWithIndexes: Use indexed columns in WHERE condition on large tables
Critical	Regex Injection	OPT.ABAP.SEC.RegexInjection	RegexInjection: Prevent denial of service attack through malicious regular expression ('Regex Injection')
Medium	Replace If With Case	OPT.ABAP.APFR.ReplacelfWithCase	ReplacelfWithCase: Replace IF ... ENDIF with CASE ... ENDCASE when possible
High	Rfc Callback Attack	OPT.ABAP.SEC.RfcCallbackAttack	RfcCallbackAttack: RFC call without callback attack protection
Critical	Rfc Destination Injection	OPT.ABAP.SEC.RfcDestinationInjection	RfcDestinationInjection: Destination injection in RFC call
Medium	Security Select Tables	OPT.ABAP.ASR.SecuritySelectTables	SecuritySelectTables: Avoid queries on sensitive tables from ABAP code
Low	Select Into Instead Of Select Appending	OPT.ABAP.APFR.SelectIntoInsteadOfSelectAppending	SelectIntoInsteadOfSelectAppending: Use SELECT + INTO instead of SELECT APPENDING
Low	Select Into Table Instead Of Select End Select	OPT.ABAP.APFR.SelectIntoTableInsteadOfSelectEndSelect	SelectIntoTableInsteadOfSelectEndSelect: Do not use SELECT...ENDSELECT for loading an internal table
Low	Set Get Sys Param	OPT.ABAP.APBR.SetGetSysParam	SetGetSysParam: Use existing parameters in the system
High	Sort Before Removing Duplicates	OPT.ABAP.RELIABILITY.SortBeforeRemovingDuplicates	SortBeforeRemovingDuplicates: Sort internal tables before removing duplicates
Low	Sort Instead Of Order By	OPT.ABAP.ASR.SortInsteadOfOrderBy	SortInsteadOfOrderBy: Use ABAP SORT instead of ORDER BY in SELECT

Severity	Contrast rule	Engine rule ID	Description
Critical	Sort Stmt In A Loop	OPT.ABAP.EFFICIENCY.SortStmtInALoop	SortStmtInALoop: Avoid declaring SORT statements inside a LOOP
Critical	SQL Injection	OPT.ABAP.SEC.SqlInjection	SqlInjection: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
Medium	Stand Comment Form	OPT.ABAP.ADR.StandCommentForm	StandCommentForm: In the FORM it is advisable to use the comments with standard format
High	Submit Report	OPT.ABAP.AGR.SubmitReport	SubmitReport: Avoid too many SUBMIT calls in a report
Low	Submit Report Type1	OPT.ABAP.APBR.SubmitReportType1	SubmitReportType1: Avoid call reports that are not type 1
Critical	Submit Stmt In A Loop	OPT.ABAP.EFFICIENCY.SubmitStmtInALoop	SubmitStmtInALoop: Avoid declaring SUBMIT statements inside a LOOP
Low	Subroutine Definitions After Events	OPT.ABAP.AGR.SubroutineDefinitionsAfterEvents	SubroutineDefinitionsAfterEvents: Every subroutine should be declared after events
Low	Suggest Append Lines Instead Of Append	OPT.ABAP.APFR.SuggestAppendLinesInsteadOfAppend	SuggestAppendLinesInsteadOfAppend: For efficiency, use APPEND LINES OF instead of APPEND
Medium	Suggest Others Exceptions	OPT.ABAP.APBR.SuggestOthersExceptions	SuggestOthersExceptions: Call with no OTHERS option in EXCEPTIONS
High	Suggest Select Where	OPT.ABAP.ASR.SuggestSelectWhere	SuggestSelectWhere: SELECT without WHERE
Low	Suggest Typed Parameters	OPT.ABAP.AGR.SuggestTypedParameters	SuggestTypedParameters: Avoid procedures with untyped or too generic type parameters
High	Suggest While Instead Of Do	OPT.ABAP.APFR.SuggestWhileInsteadOfDo	SuggestWhileInsteadOfDo: Use WHILE instead of unconditional DO loops
Low	Too Many Attributes In A Class	OPT.ABAP.MAINT.TooManyAttributesInAClass	TooManyAttributesInAClass: Avoid declaring too many class attributes
Low	Too Many Database Operations In Block	OPT.ABAP.MAINT.TooManyDatabaseOperationsInBlock	TooManyDatabaseOperationsInBlock: Avoid too many database operations in a behavioral unit
Low	Too Many Lines By File	OPT.ABAP.MAINT.TooManyLinesByFile	TooManyLinesByFile: Avoid files with too many lines
Medium	Too Nested Macro Calls	OPT.ABAP.MAINT.TooNestedMacroCalls	TooNestedMacroCalls: Avoid too nested macro calls
High	Uncaught Exception In Rfc Call	OPT.ABAP.RELIABILITY.UncaughtExceptionInRfcCall	UncaughtExceptionInRfcCall: Uncaught exception in RFC call
Medium	Unicode Programs	OPT.ABAP.ADR.UnicodePrograms	UnicodePrograms: The code must guard compatibility with UNICODE
High	Update Dbtable Without Where	OPT.ABAP.RELIABILITY.UpdateDbtableWithoutWhere	UpdateDbtableWithoutWhere: UPDATE on database table without WHERE
High	Update Delete Without Where	OPT.ABAP.EFFICIENCY.UpdateDeleteWithoutWhere	UpdateDeleteWithoutWhere: UPDATE / DELETE without WHERE
Info	Usages Of Sy Sysid	OPT.ABAP.SEC.UsagesOfSySysid	UsagesOfSySysid: Usage of sy-sysid (informative)

Severity	Contrast rule	Engine rule ID	Description
Info	Usages Of Sy Uname	OPT.ABAP.SEC.UsagesOfSyUname	UsagesOfSyUname: Usage of sy- uname (informative)
Medium	Use Attributes Controller Class	OPT.ABAP.AWD.UseAttributesControllerClass	UseAttributesControllerClass: Web Dynpro controller classes must have the attributes WDTTHIS and WDCONTEXT
Medium	Use Class Based Exceptions	OPT.ABAP.MAINT.UseClassBasedExceptions	UseClassBasedExceptions: Use class-based exception handling
High	Use For All Entries	OPT.ABAP.APFR.UseForAllEntries	UseForAllEntries: Ensure that internal table in FOR ALL ENTRIES clause is not empty
Medium	Use Local Instead Of Tables In Subroutines	OPT.ABAP.AGR.UseLocalInsteadOfTablesInSubroutines	UseLocalInsteadOfTablesInSubroutines: Do not declare TABLES/NODES in procedures, modules or event blocks
High	Use Read With Binary Search If With Key	OPT.ABAP.APFR.UseReadWithBinarySearchIfWithKey	UseReadWithBinarySearchIfWithKey: Use BINARY SEARCH with WITH KEY when reading tables
High	Warn Scroll	OPT.ABAP.AWD.WarnScroll	WarnScroll: Avoid Web Dynpro elements with scrollingMode {}
Low	Warn Without Adobe Print Form	OPT.ABAP.ADR.WarnWithoutAdobePrintForm	WarnWithoutAdobePrintForm: Warning if Print Forms is not in use Adobe as form
High	Weak Hash Algorithm	OPT.ABAP.SEC.WeakHashAlgorithm	WeakHashAlgorithm: Weak cryptographic hashes cannot guarantee data integrity
Low	Where Clauses Without Not Or	OPT.ABAP.ASR.WhereClausesWithoutNotOr	WhereClausesWithoutNotOr: Avoid NOT / OR operators inside WHERE clauses in database operations
Low	Working With Report	OPT.ABAP.AGR.WorkingWithReport	WorkingWithReport: Avoid dynamic operations (READ, INSERT or DELETE) with reports
Low	Working With Text Pool	OPT.ABAP.AGR.WorkingWithTextPool	WorkingWithTextPool: Avoid read, insert or delete textpools

ActionScript Scan rules

Contrast Scan supports these rules for ActionScript.

Severity	Contrast rule	Engine rule ID	Description
Critical	Avoid Declaring Vars Without Var	OPT.ACTIONSCRIPT.EST_ACTIONSCRIPT.AvoidDeclaringVarsWithoutVar	AvoidDeclaringVarsWithoutVar: Avoid declaring vars without reserve word VAR before
Critical	End Sentences With Semicolon	OPT.ACTIONSCRIPT.EST_ACTIONSCRIPT.EndSentencesWithSemicolon	EndSentencesWithSemicolon: Avoid sentences without semicolon at the end
Critical	Avoid Loop With Empty Body	OPT.ACTIONSCRIPT.GEN_ACTIONSCRIPT.AvoidLoopWithEmptyBody	AvoidLoopWithEmptyBody: Avoid empty loops
Critical	Comp Life Cycle	OPT.ACTIONSCRIPT.GEN_ACTIONSCRIPT.CompLifeCycle	CompLifeCycle: Avoid overriding certain methods
Critical	No Keywords As Identifiers	OPT.ACTIONSCRIPT.NOM_ACTIONSCRIPT.NoKeywordsAsIdentifiers	NoKeywordsAsIdentifiers: Do not use keywords as identifiers

Severity	Contrast rule	Engine rule ID	Description
Critical	Avoid Declaring Inside Loops	OPT.ACTIONSCRIPT.PER_ACTIONSCRIPT.AvoidDeclaringInsideLoops	AvoidDeclaringInsideLoops: Avoid declaring variables inside loops
Critical	Avoid Large Classes	OPT.ACTIONSCRIPT.PER_ACTIONSCRIPT.AvoidLargeClasses	AvoidLargeClasses: Avoid large classes
Critical	Avoid Large Constructors	OPT.ACTIONSCRIPT.PER_ACTIONSCRIPT.AvoidLargeConstructors	AvoidLargeConstructors: Avoid large constructors
Critical	Avoid Large Functions	OPT.ACTIONSCRIPT.PER_ACTIONSCRIPT.AvoidLargeFunctions	AvoidLargeFunctions: Avoid large functions
Critical	Avoid Many Conditionals	OPT.ACTIONSCRIPT.PER_ACTIONSCRIPT.AvoidManyConditionals	AvoidManyConditionals: Avoid many conditional sentences in the code
Critical	Avoid Mutiple Return	OPT.ACTIONSCRIPT.PER_ACTIONSCRIPT.AvoidMutipleReturn	AvoidMutipleReturn: Avoid multiple RETURN statements inside a function
Critical	Avoid Nested Loops	OPT.ACTIONSCRIPT.PER_ACTIONSCRIPT.AvoidNestedLoops	AvoidNestedLoops: Avoid using nested loops
Critical	Declaring Array	OPT.ACTIONSCRIPT.PER_ACTIONSCRIPT.DeclaringArray	DeclaringArray: Declaring Arrays using new array instead of []
Critical	No Use Set Style	OPT.ACTIONSCRIPT.PER_ACTIONSCRIPT.NoUseSetStyle	NoUseSetStyle: Avoid 'setStyle()' method
Critical	Too Many Parameters In Function	OPT.ACTIONSCRIPT.PER_ACTIONSCRIPT.TooManyParametersInFunction	TooManyParametersInFunction: Too many parameters in a function
Critical	Type Everything	OPT.ACTIONSCRIPT.PER_ACTIONSCRIPT.TypeEverything	TypeEverything: Every var, function, parameter and constant must be declared along with a type
High	Use Space Between Operators	OPT.ACTIONSCRIPT.EST_ACTIONSCRIPT.UseSpaceBetwenOperators	UseSpaceBetwenOperators: Put a single space around operators
High	Avoid Assignment in Condition	OPT.ACTIONSCRIPT.GEN_ACTIONSCRIPT.AvoidAssignmentinCondition	AvoidAssignmentinCondition: Avoid a assignment operation in condition statements
High	Avoid Empty Functions	OPT.ACTIONSCRIPT.GEN_ACTIONSCRIPT.AvoidEmptyFunctions	AvoidEmptyFunctions: Avoid declaring empty functions
High	Avoid For With External Control Vars	OPT.ACTIONSCRIPT.GEN_ACTIONSCRIPT.AvoidForWithExternalControlVars	AvoidForWithExternalControlVars: Avoid using external variables to loop control in FOR statements
High	Avoid Unary Ops In Assign	OPT.ACTIONSCRIPT.GEN_ACTIONSCRIPT.AvoidUnaryOpsInAssign	AvoidUnaryOpsInAssign: Avoid ternary operator in assignment statements (?:)
High	Avoid Unused Function	OPT.ACTIONSCRIPT.GEN_ACTIONSCRIPT.AvoidUnusedFunction	AvoidUnusedFunction: Avoid having declared unused functions
High	Avoid Unused Function Parameter	OPT.ACTIONSCRIPT.GEN_ACTIONSCRIPT.AvoidUnusedFunctionParameter	AvoidUnusedFunctionParameter: Detect function parameter not used
High	Avoid Unused Local Var	OPT.ACTIONSCRIPT.GEN_ACTIONSCRIPT.AvoidUnusedLocalVar	AvoidUnusedLocalVar: Detect local vars unused
High	If With Out Block	OPT.ACTIONSCRIPT.GEN_ACTIONSCRIPT.IfWithOutBlock	IfWithOutBlock: Avoid using IF statements with empty body
High	No Update Loop In For Body	OPT.ACTIONSCRIPT.GEN_ACTIONSCRIPT.NoUpdateLoopInForBody	NoUpdateLoopInForBody: Avoid to update control var inside FOR loop body
High	Unused Constant	OPT.ACTIONSCRIPT.GEN_ACTIONSCRIPT.UnusedConstant	UnusedConstant: Avoid unused private constant declarations
High	Avoid Numbers As Control Var	OPT.ACTIONSCRIPT.PER_ACTIONSCRIPT.AvoidNumbersAsControlVar	AvoidNumbersAsControlVar: Avoid declaring control variables as Numbers
High	Avoid Popup Windows	OPT.ACTIONSCRIPT.PER_ACTIONSCRIPT.AvoidPopupWindows	AvoidPopupWindows: Use javascript to open pop-up windows

Severity	Contrast rule	Engine rule ID	Description
Info	Document Every Function	OPT.ACTIONSCRIPT.DOC_ACTIONSCRIPT.DocumentEveryFunction	DocumentEveryFunction: Provide comments for functions
Low	Code Document Percentage	OPT.ACTIONSCRIPT.DOC_ACTIONSCRIPT.CodeDocumentPercentage	CodeDocumentPercentage: Avoid files with a very low comment/ code ratio
Low	Avoid Switch Many Cases	OPT.ACTIONSCRIPT.GEN_ACTIONSCRIPT.AvoidSwitchManyCases	AvoidSwitchManyCases: Avoid too many CASE clauses in SWITCH statement
Medium	Class Naming Pattern	OPT.ACTIONSCRIPT.NOM_ACTIONSCRIPT.ClassNamingPattern	ClassNamingPattern: Class names must comply with the convention
Medium	Constant Naming Pattern	OPT.ACTIONSCRIPT.NOM_ACTIONSCRIPT.ConstantNamingPattern	ConstantNamingPattern: Constant names must comply with the convention
Medium	Function Naming Pattern	OPT.ACTIONSCRIPT.NOM_ACTIONSCRIPT.FunctionNamingPattern	FunctionNamingPattern: Function names must comply with the convention
Medium	Interface Naming Pattern	OPT.ACTIONSCRIPT.NOM_ACTIONSCRIPT.InterfaceNamingPattern	InterfaceNamingPattern: Interface names must comply with the convention
Medium	Package Naming Pattern	OPT.ACTIONSCRIPT.NOM_ACTIONSCRIPT.PackageNamingPattern	PackageNamingPattern: Package names must comply with the convention
Medium	Variable Naming Pattern	OPT.ACTIONSCRIPT.NOM_ACTIONSCRIPT.VariableNamingPattern	VariableNamingPattern: Variable names must follow the conventions
Medium	Misuse Dynamic Filter	OPT.ACTIONSCRIPT.PER_ACTIONSCRIPT.MisuseDynamicFilter	MisuseDynamicFilter: Avoid declaring unused filters

ASP.NET Scan rules

Contrast Scan supports these rules for ASP.NET.

Severity	Contrast rule	Engine rule ID	Description
Critical	Dangerous App Setting	OPT.ASPNET.DangerousAppSetting	DangerousAppSetting: Dangerous application setting
Critical	Too Broad CORS Policy	OPT.ASPNET.TooBroadCORSPolicy	TooBroadCORSPolicy: CORS policy (Cross-Origin Resource Sharing) too broad
Critical	CBII	OPT.ASPNET.CBII	CBII: Use code-behind files
Critical	Respect MVC	OPT.ASPNET.CBNC	CBNC: Respect MVC
Critical	Enable View State Mac	OPT.ASPNET.EnableViewStateMac	EnableViewStateMac: Do not set EnableViewStateMac {'OWASP-2021': ['A5'], 'PCI-DSS': ['6.5.1']}
Critical	Cross Site Scripting	OPT.ASPNET.CrossSiteScripting	CrossSiteScripting: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
High	Avoid Enabled Debug Mode	OPT.ASPNET.AvoidEnabledDebugMode	AvoidEnabledDebugMode: ASP.NET Misconfiguration: Creating Debug Binary
High	Clickjacking Protection	OPT.ASPNET.ClickjackingProtection	ClickjackingProtection: No clickjacking protection configured
High	Re DoS In Regular Expression Validator	OPT.ASPNET.ReDoSInRegularExpressionValidator	ReDoSInRegularExpressionValidator: Regular expression in RegularExpressionValidator may be used for denial of service

Severity	Contrast rule	Engine rule ID	Description
High	Session Hijacking Misconfiguration	OPT.ASPNET.SessionHijackingMisconfiguration	SessionHijackingMisconfiguration: A misconfiguration makes easier performing Session hijacking attacks
High	Trace Enabled	OPT.ASPNET.TraceEnabled	TraceEnabled: Trace information enabled and remotely accessible
High	Unprotected Roles In Cookies	OPT.ASPNET.UnprotectedRolesInCookies	UnprotectedRolesInCookies: Unprotected roles in cookies
High	WCF Audit Misconfiguration	OPT.ASPNET.WCFAuditMisconfiguration	WCFAuditMisconfiguration: Audit of security events misconfiguration in WCF
High	WCF Transport Security	OPT.ASPNET.WCFTransportSecurity	WCFTransportSecurity: Do not use transport security mode in WCF
High	Avoid Impersonation	OPT.ASPNET.AvoidImpersonation	AvoidImpersonation: Avoid impersonation in ASP.Net configuration
High	HTTP Verb Tampering	OPT.ASPNET.HTTPVerbTampering	HTTPVerbTampering: Misconfiguration in authorization rules allowing HTTP Verb Tampering
High	Header Validation Misconfiguration	OPT.ASPNET.HeaderValidationMisconfiguration	HeaderValidationMisconfiguration: Unvalidated data in HTTP response header ('HTTP Response Splitting')
High	Path Relative Stylesheet Import	OPT.ASPNET.PathRelativeStylesheetImport	PathRelativeStylesheetImport: Path-Relative Stylesheet Import.
High	Target Blank Vulnerability	OPT.ASPNET.TargetBlankVulnerability	TargetBlankVulnerability: Improper Neutralization of links to external sites
High	Credentials Misconfiguration	OPT.ASPNET.CredentialsMisconfiguration	CredentialsMisconfiguration: Password exposure in Web.config file
High	Prevent MIME Sniffing	OPT.ASPNET.PreventMIMESniffing	PreventMIMESniffing: Prevent MIME sniffing
Low	Don't use masterpage files	OPT.ASPNET.MP	MP: Don't use masterpage files
Low	Form Without Captcha	OPT.ASPNET.FormWithoutCaptcha	FormWithoutCaptcha: Form without CAPTCHA
Medium	Authentication Forms Without SSL	OPT.ASPNET.AuthenticationFormsWithoutSSL	AuthenticationFormsWithoutSSL: If authentication is through Forms enable the sending of information through SSL
Medium	Avoid Disabled Validate Request	OPT.ASPNET.AvoidDisabledValidateRequest	AvoidDisabledValidateRequest: The value of ValidateRequest in pages must be set to true to prevent code injection attacks
Medium	Avoid Disabled Validate Request Config	OPT.ASPNET.AvoidDisabledValidateRequestConfig	AvoidDisabledValidateRequestConfig: The validateRequest attribute value should be true to prevent code injection attacks
Medium	Avoid Send Cookies Unencrypted HTTP	OPT.ASPNET.AvoidSendCookiesUnencryptedHTTP	AvoidSendCookiesUnencryptedHTTP: Sending of cookies should be enabled only by HTTP
Medium	Avoid Send Cookies Without SSL	OPT.ASPNET.AvoidSendCookiesWithoutSSL	AvoidSendCookiesWithoutSSL: Send Cookies using SSL
Medium	Directory Browsing	OPT.ASPNET.DirectoryBrowsing	DirectoryBrowsing: Directory Browsing enabled
Medium	Forms Authentication Timeout	OPT.ASPNET.FormsAuthenticacionTimeout	FormsAuthenticacionTimeout: Set expiration timeout for authentication cookies
Medium	Service Metadata Visibility	OPT.ASPNET.ServiceMetadataVisibility	ServiceMetadataVisibility: Service metadata exposure

Severity	Contrast rule	Engine rule ID	Description
Medium	WCF Avoid Enabled Debug	OPT.ASPNET.WCFAvoidEnabledDebug	WCFAvoidEnabledDebug: Avoid enabling WCF debug information
Medium	Avoid Empty Files	OPT.ASPNET.AvoidEmptyFiles	AvoidEmptyFiles: Avoid empty files
Medium	Avoid Enabled View State Mode Page	OPT.ASPNET.AvoidEnabledViewStateModePage	AvoidEnabledViewStateModePage: Avoid enable ViewStateMode at page level
Medium	Avoid Use Style Of Controls	OPT.ASPNET.AvoidUseStyleOfControls	AvoidUseStyleOfControls: Avoid using styles by tags attributes. Use the CssClass attribute
Medium	Enable Custom Error Page	OPT.ASPNET.EnableCustomErrorPage	EnableCustomErrorPage: ASP.NET Misconfiguration: Missing Custom Error Page
Medium	Avoid Content Delivery Network	OPT.ASPNET.AvoidContentDeliveryNetwork	AvoidContentDeliveryNetwork: Do not use Content Delivery Network (CDN) for JavaScript code
Medium	Specify Integrity Attribute	OPT.ASPNET.SpecifyIntegrityAttribute	SpecifyIntegrityAttribute: Specify a integrity attribute on the <script> and <link> elements
Medium	Credentials In Connection String	OPT.ASPNET.CredentialsInConnectionString	CredentialsInConnectionString: Insufficiently protected credentials in connection strings
Medium	Persist Security Info True	OPT.ASPNET.PersistSecurityInfoTrue	PersistSecurityInfoTrue: Persist Security Info enabled in connection strings

ASP Scan rules

Contrast Scan supports these rules for ASP.

Severity	Contrast rule	Engine rule ID	Description
Critical	ASP Avoid Page Transfer	OPT.ASP.ASP_FMT.ASP_AvoidPageTransfer	ASP_AvoidPageTransfer: Avoid redirections between ASP pages
Critical	ASP Naming Convention	OPT.ASP.ASP_NAM.ASP_NamingConvention	ASP_NamingConvention: Identifier names (variables, constants, procedures) in ASP must follow naming standard
Critical	ASP Page Name	OPT.ASP.ASP_NAM.ASP_PageName	ASP_PageName: ASP page names must follow a naming standard
Critical	ASP SQL Injection	OPT.ASP.ASP_SEC.ASP_SqlInjection	ASP_SqlInjection: Checks for SQL injection vulnerabilities
High	ASP Avoid Document Write	OPT.ASP.ASP_FMT.ASP_AvoidDocumentWrite	ASP_AvoidDocumentWrite: Insertion of HTML code from ASP file including document.write commands
High	ASP No Java Script	OPT.ASP.ASP_FMT.ASP_NoJavaScript	ASP_NoJavaScript: Do not include directly JavaScript functions in ASP pages
High	ASP Use Option Explicit	OPT.ASP.ASP_FMT.ASP_UseOptionExplicit	ASP_UseOptionExplicit: Option Explicit must be used in every ASP page
High	ASP Use Stored Procedures	OPT.ASP.ASP_UseStoredProcedures	ASP_UseStoredProcedures: ASP pages must perform database operations using stored procedures
Info	Use Header Comment	OPT.ASP.ASP_DOC.UseHeaderComment	UseHeaderComment: A proper comment must be placed at the top of every ASP page
Info	ASP No Use HTML Comments	OPT.ASP.ASP_FMT.ASP_NoUseHTMLComments	ASP_NoUseHTMLComments: Do not use HTML comments

Severity	Contrast rule	Engine rule ID	Description
Low	ASP Avoid Styles	OPT.ASP.ASP_FMT.ASP_AvoidStyles	ASP_AvoidStyles: Do not encode style information in HTML code in ASP pages
Medium	Avoid Data Base Access	OPT.ASP.ASP_DB.AvoidDataBaseAccess	AvoidDataBaseAccess: Avoid database access from ASP pages
Medium	ASP Avoid Duplicate Files	OPT.ASP.ASP_FMT.ASP_AvoidDuplicateFiles	ASP_AvoidDuplicateFiles: Duplicated ASP pages in different places
Medium	ASP Avoid Empty Pages	OPT.ASP.ASP_FMT.ASP_AvoidEmptyPages	ASP_AvoidEmptyPages: Do not add empty ASP pages to the site
Medium	ASP Iframes Without Src	OPT.ASP.ASP_FMT.ASP_IframesWithoutSrc	ASP_IframesWithoutSrc: No dejar el elemento iframes sin atributo src
Medium	ASP No Commented Java Script	OPT.ASP.ASP_FMT.ASP_NoCommentedJavaScript	ASP_NoCommentedJavaScript: Do not leave commented JavaScript code in ASP page

C# Scan rules

Contrast Scan supports these rules for C#.

Severity	Contrast rule	Engine rule ID	Description
Critical	Too Much Origins Allowed	OPT.CSHARP.TooMuchOriginsAllowed	TooMuchOriginsAllowed: CORS policy (Cross origin resource sharing) too broad
Critical	Avoid Exception Throwing In Binary Operators	OPT.CSHARP.Csharp.AvoidExceptionThrowingInBinaryOperators	AvoidExceptionThrowingInBinaryOperators: The override of the Equality binary operator should not throw an exception
Critical	Avoid Exception Throwing In Dispose Method	OPT.CSHARP.Csharp.AvoidExceptionThrowingInDisposeMethod	AvoidExceptionThrowingInDisposeMethod: Dispose method should not throw exception
Critical	Avoid Exception Throwing In Equals Clause	OPT.CSHARP.Csharp.AvoidExceptionThrowingInEqualsClause	AvoidExceptionThrowingInEqualsClause: The override of Equals method should not throw exception
Critical	Avoid Exception Throwing In Get Hash Code	OPT.CSHARP.Csharp.AvoidExceptionThrowingInGetHashCode	AvoidExceptionThrowingInGetHashCode: The override of the GetHashCode method should not throw an exception
Critical	Dispose Objects Before Losing Scope	OPT.CSHARP.Csharp.DisposeObjectsBeforeLosingScope	DisposeObjectsBeforeLosingScope: Dispose objects before losing scope
Critical	Do Not Dispose Objects Multiple Times	OPT.CSHARP.Csharp.DoNotDisposeObjectsMultipleTimes	DoNotDisposeObjectsMultipleTimes: Possibility of multiple calls to Dispose over an object
Critical	Do Not Use Idle Process Priority	OPT.CSHARP.Csharp.DoNotUseIdleProcessPriority	DoNotUseIdleProcessPriority: Do not use idle process priority
Critical	Mark I Serializable Types With Serializable	OPT.CSHARP.Csharp.MarkISerializableTypesWithSerializable	MarkISerializableTypesWithSerializable: Specify the Serializable attribute in ISerializable class
Critical	Mark Windows Forms Entry Points With Sta Thread	OPT.CSHARP.Csharp.MarkWindowsFormsEntryPointsWithStaThread	MarkWindowsFormsEntryPointsWithStaThread: Mark Windows Forms entry points with STA Thread

Severity	Contrast rule	Engine rule ID	Description
Critical	Overriding Equal And Distinct Operators	OPT.CSHARP.Csharp.OverridingEqualAndDistinctOperators	OverridingEqualAndDistinctOperators: Any that overloads operator Code Quality [] {}
Critical	Overriding Equals And Get Hash Code	OPT.CSHARP.Csharp.OverridingEqualsAndGetHashCode	OverridingEqualsAndGetHashCode: Any ty that overrides GetHashCode method should also override Equals method
Critical	Null Dereference	OPT.CSHARP.NullDereference	NullDereference: NULL Pointer Dereference
Critical	Path Traversal	OPT.CSHARP.PathTraversal	PathTraversal: Avoid non-neutralized user-controlled input composed in a pathname to resource
Critical	Reference Self Assigned	OPT.CSHARP.ReferenceSelfAssigned	ReferenceSelfAssigned: Reference is self assigned in assignment statement
Critical	Same Conditional In If Else If	OPT.CSHARP.SameConditionalInIfElseIf	SameConditionalInIfElseIf: Duplicated conditional in if-else if statements
Critical	Same Implementation In Conditional	OPT.CSHARP.SameImplementationInConditional	SameImplementationInConditional: Same implementation for different branches in a conditional statement
Critical	Same Subexpression In Logical Expression	OPT.CSHARP.SameSubexpressionInLogicalExpression	SameSubexpressionInLogicalExpression: Duplicated subexpression in logical expression
Critical	Accessibility Subversion Rule	OPT.CSHARP.SEC.AccessibilitySubversionRule	AccessibilitySubversionRule: .Net access restriction subverted (Reflection)
Critical	Anonymous Ldap Bind	OPT.CSHARP.SEC.AnonymousLdapBind	AnonymousLdapBind: Access Control - Anonymous LDAP Bind
Critical	Dangerous File Upload	OPT.CSHARP.SEC.DangerousFileUpload	DangerousFileUpload: Unrestricted Upload File with Dangerous Type
Critical	Code Injection	OPT.CSHARP.CodeInjection	CodeInjection: Improper Control of Generation of Code ('Code Injection')
Critical	Code Injection With Deserialization	OPT.CSHARP.CodeInjectionWithDeserialization	CodeInjectionWithDeserialization: Dynamic code injection during object deserialization
Critical	Command Injection	OPT.CSHARP.CommandInjection	CommandInjection: Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
Critical	Static Database Connection	OPT.CSHARP.SEC.StaticDatabaseConnection	StaticDatabaseConnection: Static database connection / session
Critical	Temporary Files Left	OPT.CSHARP.SEC.TemporaryFilesLeft	TemporaryFilesLeft: Temporary files not deleted
Critical	Cross Site Scripting	OPT.CSHARP.CrossSiteScripting	CrossSiteScripting: Improper Neutralization of Input During Web Page Generation ('Cross Site Scripting')
Critical	DoS Regexp	OPT.CSHARP.DoSRegexp	DoSRegexp: Prevent denial of service attack through malicious regular expression
Critical	Ldap Injection	OPT.CSHARP.LdapInjection	LdapInjection: Avoid non-neutralized user-controlled input in LDAP search filters
Critical	Connection String Parameter Pollution	OPT.CSHARP.SEC.ConnectionStringParameterPollution	ConnectionStringParameterPollution: Connection string polluted with untrusted input
Critical	Http Parameter Pollution	OPT.CSHARP.SEC.HttpParameterPollution	HttpParameterPollution: HTTP parameter pollution (HPP)
Critical	Http Splitting Rule	OPT.CSHARP.SEC.HttpSplittingRule	HttpSplittingRule: Improper neutralization of CR/LF Sequences in HTTP headers
Critical	Mail Command Injection	OPT.CSHARP.SEC.MailCommandInjection	MailCommandInjection: Mail Command Injection

Severity	Contrast rule	Engine rule ID	Description
Critical	No SQL Injection	OPT.CSHARP.SEC.NoSQLInjection	NoSQLInjection: Improper neutralization of special elements in data query logic (NoSQL injection)
Critical	Wrong Lock Usage	OPT.CSHARP.WrongLockUsage	WrongLockUsage: Wrong lock acquisition/release
Critical	Process Control	OPT.CSHARP.SEC.ProcessControl	ProcessControl: Do not load executables or libraries from untrusted sources
Critical	Registry Manipulation	OPT.CSHARP.SEC.RegistryManipulation	RegistryManipulation: Registry manipulation
Critical	Server Side Request Forgery	OPT.CSHARP.ServerSideRequestForgery	ServerSideRequestForgery: Server-Side Request Forgery (SSRF)
Critical	SQL Injection	OPT.CSHARP.SqlInjection	SqlInjection: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
Critical	Stored Cross Site Scripting	OPT.CSHARP.StoredCrossSiteScripting	StoredCrossSiteScripting: Web content generation from improper sanitized database data and escaped output (Stored Cross-site Scripting, XSS)
Critical	MVC Non Action Public Methods	OPT.CSHARP.MVCNonActionPublicMethods	MVCNonActionPublicMethods: Protect public methods that are not action methods in controllers
Critical	Weak Cryptographic Hash	OPT.CSHARP.WeakCryptographicHash	WeakCryptographicHash: Weak cryptographic hash
Critical	Weak Key Size	OPT.CSHARP.WeakKeySize	WeakKeySize: Weak cryptography, insufficient key length
Critical	Weak Symmetric Encryption Algorithm	OPT.CSHARP.WeakSymmetricEncryptionAlgorithm	WeakSymmetricEncryptionAlgorithm: Weak symmetric encryption algorithm
Critical	Weak Symmetric Encryption Mode Of Operation	OPT.CSHARP.WeakSymmetricEncryptionModeOfOperation	WeakSymmetricEncryptionModeOfOperation: Do not use weak modes of operation with symmetric encryption
High	Plaintext Storage In A Cookie	OPT.CSHARP.PlaintextStorageInACookie	PlaintextStorageInACookie: Cleartext Storage of Sensitive Information in a Cookie
High	Abstract Types Should Not Have Constructors	OPT.CSHARP.Csharp.AbstractTypesShouldNotHaveConstructors	AbstractTypesShouldNotHaveConstructors: Public abstract class with public constructors
High	Avoid Com Parameterized Constructors	OPT.CSHARP.Csharp.AvoidComParameterizedConstructors	AvoidComParameterizedConstructors: COM does not support parameterized constructors
High	Avoid Com Static Methods	OPT.CSHARP.Csharp.AvoidComStaticMethods	AvoidComStaticMethods: COM does not support static methods
High	Avoid Empty Catch Block	OPT.CSHARP.Csharp.AvoidEmptyCatchBlock	AvoidEmptyCatchBlock: Avoid empty catch blocks
High	Avoid Excessive Complexity	OPT.CSHARP.Csharp.AvoidExcessiveComplexity	AvoidExcessiveComplexity: A method has excessive cyclomatic complexity
High	Avoid Excessive Locals	OPT.CSHARP.Csharp.AvoidExcessiveLocals	AvoidExcessiveLocals: To avoid to declare excessive local variables
High	Avoid Floating Point Equality	OPT.CSHARP.Csharp.AvoidFloatingPointEquality	AvoidFloatingPointEquality: Do not perform (in)equality operations over floating point variables
High	Avoid Inconditional Recursive Invocation	OPT.CSHARP.Csharp.AvoidInconditionalRecursiveInvocation	AvoidInconditionalRecursiveInvocation: Avoid recursive calls without a precondition

Severity	Contrast rule	Engine rule ID	Description
High	Avoid Out Parameters	OPT.CSHARP.Csharp.AvoidOutParameters	AvoidOutParameters: Public or protected methods must avoid ref/out parameters
High	Avoid Overloads In Com Visible Interfaces	OPT.CSHARP.Csharp.AvoidOverloadsInComVisibleInterfaces	AvoidOverloadsInComVisibleInterfaces: COM visible interface declares overloaded methods
High	Avoid Uncalled Private Code	OPT.CSHARP.Csharp.AvoidUncalledPrivateCode	AvoidUncalledPrivateCode: Avoid uncalled private code
High	Avoid Unused Private Fields	OPT.CSHARP.Csharp.AvoidUnusedPrivateFields	AvoidUnusedPrivateFields: Avoid unused private fields
High	Declare Event Handlers Correctly	OPT.CSHARP.Csharp.DeclareEventHandlersCorrectly	DeclareEventHandlersCorrectly: Declare Event Handlers Correctly
High	Do Not Assume Int Ptr Size Rule	OPT.CSHARP.Csharp.DoNotAssumeIntPtrSizeRule	DoNotAssumeIntPtrSizeRule: Do not downcast IntPtr or UIntPtr into a 32-bit or smaller value
High	Do Not Declare Static Members On Generic Types	OPT.CSHARP.Csharp.DoNotDeclareStaticMembersOnGenericTypes	DoNotDeclareStaticMembersOnGenericTypes: Do not declare static members on generic types
High	Do Not Declare Visible Instance Fields	OPT.CSHARP.Csharp.DoNotDeclareVisibleInstanceFields	DoNotDeclareVisibleInstanceFields: Avoid externally visible instance fields
High	Do Not Lock On This Or Types	OPT.CSHARP.Csharp.DoNotLockOnThisOrTypes	DoNotLockOnThisOrTypes: Do not lock on this, types, or string literals
High	Do Not Pass Types By Reference	OPT.CSHARP.Csharp.DoNotPassTypesByReference	DoNotPassTypesByReference: Do not pass types by reference
High	Exceptions Should Be Public	OPT.CSHARP.Csharp.ExceptionsShouldBePublic	ExceptionsShouldBePublic: Exceptions should be public
High	I Comparable Overriding Methods	OPT.CSHARP.Csharp.IComparableOverridingMethods	IComparableOverridingMethods: Override methods on comparable types
High	Identifiers Should Not Match Keywords	OPT.CSHARP.Csharp.IdentifiersShouldNotMatchKeywords	IdentifiersShouldNotMatchKeywords: The identifiers must not be reserved words
High	Initialize Reference Type Static Fields Inline	OPT.CSHARP.Csharp.InitializeReferenceTypeStaticFieldsInline	InitializeReferenceTypeStaticFieldsInline: Initialize reference type static fields inline
High	Mark Boolean P Invoke Arguments With Marshal As	OPT.CSHARP.Csharp.MarkBooleanPInvokeArgumentsWithMarshalAs	MarkBooleanPInvokeArgumentsWithMarshalAs: Mark boolean P/Invoke arguments with MarshalAs attribute
High	Max Methods	OPT.CSHARP.Csharp.MaxMethods	MaxMethods: Maximum allowed number of methods
High	Move P Invokes To Native Methods Class	OPT.CSHARP.Csharp.MovePInvokesToNativeMethodsClass	MovePInvokesToNativeMethodsClass: Move P Invokes to NativeMethods class
High	Nested Namespace Dependency	OPT.CSHARP.Csharp.NestedNamespaceDependency	NestedNamespaceDependency: Type dependency in nested namespace
High	Non Constant Fields Should Not Be Visible	OPT.CSHARP.Csharp.NonConstantFieldsShouldNotBeVisible	NonConstantFieldsShouldNotBeVisible: A public or protected static field is not constant nor is it read-only

Severity	Contrast rule	Engine rule ID	Description
High	Overloading Equals Value Types	OPT.CSHARP.Csharp.OverloadingEqualsValueTypes	OverloadingEqualsValueTypes: Overload Equals() method on value types
High	Remove Unused Locals	OPT.CSHARP.Csharp.RemoveUnusedLocals	RemoveUnusedLocals: Unused local variables
High	Review Useless Control Flow Rule	OPT.CSHARP.Csharp.ReviewUselessControlFlowRule	ReviewUselessControlFlowRule: Avoid empty code blocks
High	Specify Attribute Usage	OPT.CSHARP.Csharp.SpecifyAttributeUsage	SpecifyAttributeUsage: Specify AttributeUsage on your attributes
High	Suppress Finalize Correctly	OPT.CSHARP.Csharp.SuppressFinalizeCorrectly	SuppressFinalizeCorrectly: Call GC.SuppressFinalize correctly
High	Variable Names Should Not Match Field Names	OPT.CSHARP.Csharp.VariableNamesShouldNotMatchFieldNames	VariableNamesShouldNotMatchFieldNames: Variable names should not match field names
High	Hardcoded Absolute Path	OPT.CSHARP.HardcodedAbsolutePath	HardcodedAbsolutePath: Do not hardcode absolute paths
High	Null Arg In Equals	OPT.CSHARP.NullArgInEquals	NullArgInEquals: Null argument to object.Equals()
High	Resource Leak Database	OPT.CSHARP.ResourceLeakDatabase	ResourceLeakDatabase: Unreleased database resource
High	Resource Leak Ldap	OPT.CSHARP.ResourceLeakLdap	ResourceLeakLdap: Unreleased LDAP resource
High	Resource Leak Stream	OPT.CSHARP.ResourceLeakStream	ResourceLeakStream: Unreleased stream resource
High	Resource Leak Unmanaged	OPT.CSHARP.ResourceLeakUnmanaged	ResourceLeakUnmanaged: Unreleased unmanaged resource
High	Avoid Certificate Equals	OPT.CSHARP.SEC.AvoidCertificateEquals	AvoidCertificateEquals: Never use X509Certificate.Equals() in a security context
High	Buffer Overflow	OPT.CSHARP.SEC.BufferOverflow	BufferOverflow: Potential memory corruption
High	Cookies In Security Decision	OPT.CSHARP.SEC.CookiesInSecurityDecision	CookiesInSecurityDecision: Reliance on Cookies without Validation and Integrity Checking in a Security Decision
High	Improper Authentication	OPT.CSHARP.SEC.ImproperAuthentication	ImproperAuthentication: Avoid that a user can perform actions to which he does not have access
High	Missing Standard Error Handling	OPT.CSHARP.SEC.MissingStandardErrorHandling	MissingStandardErrorHandling: Missing Standardized Error Handling Mechanism in ASP.Net
High	Cross Site Request Forgery	OPT.CSHARP.CrossSiteRequestForgery	CrossSiteRequestForgery: Cross-Site Request Forgery (CSRF)
High	Setting Manipulation	OPT.CSHARP.SEC.SettingManipulation	SettingManipulation: Setting Manipulation
High	JSON Injection	OPT.CSHARP.JSONInjection	JSONInjection: Avoid using non-neutralized user-controlled input in JSON entities
High	Unvalidated Asp Net Model	OPT.CSHARP.SEC.UnvalidatedAspNetModel	UnvalidatedAspNetModel: Unvalidated model in MVC controller
High	User Controlled SQL Primary Key	OPT.CSHARP.SEC.UserControlledSQLPrimaryKey	UserControlledSQLPrimaryKey: Avoid using user controlled Primary Key into a query
High	MVC Prevent Overposting Model Definition	OPT.CSHARP.MVCPreventOverpostingModelDefinition	MVCPreventOverpostingModelDefinition: Prevent over-posting attacks in model definition

Severity	Contrast rule	Engine rule ID	Description
High	MVC Prevent Underposting Model Composition	OPT.CSHARP.MVCPreventUnderpostingModelComposition	MVCPreventUnderpostingModelComposition: Prevent under-posting attacks in model composition
High	MVC Prevent Underposting Model Definition	OPT.CSHARP.MVCPreventUnderpostingModelDefinition	MVCPreventUnderpostingModelDefinition: Prevent under-posting attacks in model definition
High	Open Redirect	OPT.CSHARP.OpenRedirect	OpenRedirect: URL Redirection to Untrusted Site ('Open Redirect')
High	Test For NaN Correctly	OPT.CSHARP.TestForNaNCorrectly	TestForNaNCorrectly: Test for NaN correctly
High	Cross Site History Manipulation	OPT.CSHARP.SEC.CrossSiteHistoryManipulation	CrossSiteHistoryManipulation: Cross-Site History Manipulation (XSHM)
High	Transparent Methods Should Not Use Suppress Unmanaged Code Security	OPT.CSHARP.TransparentMethodsShouldNotUseSuppressUnmanagedCodeSecurity	TransparentMethodsShouldNotUseSuppressUnmanagedCodeSecurity: A transparent method should not have the attribute SuppressUnmanagedCodeSecurity
High	Use Params For Variable Arguments	OPT.CSHARP.UseParamsForVariableArguments	UseParamsForVariableArguments: A public or protected type contains a public or protected method that uses the VarArgs calling convention
High	Log Forging	OPT.CSHARP.SEC.LogForging	LogForging: Improper Output Neutralization in Logs
High	Resource Injection	OPT.CSHARP.SEC.ResourceInjection	ResourceInjection: Improper control of resource identifiers ("Resource Injection")
High	Trust Boundary Violation	OPT.CSHARP.SEC.TrustBoundaryViolation	TrustBoundaryViolation: Trust boundary violation
High	Unsafe Reflection	OPT.CSHARP.SEC.UnsafeReflection	UnsafeReflection: Use of Externally-Controlled Input to Select Classes or Code ('Unsafe Reflection')
High	XML Entity Injection	OPT.CSHARP.SEC.XMLEntityInjection	XMLEntityInjection: XML entity injection
High	XML Injection	OPT.CSHARP.XMLInjection	XMLInjection: XML Injection (aka Blind XPath Injection)
High	XPath Injection	OPT.CSHARP.XPathInjection	XPathInjection: Improper Neutralization of data within XPath Expressions ('XPath Injection')
High	XQuery Injection	OPT.CSHARP.XQueryInjection	XQueryInjection: Improper Neutralization of data within XQuery Expressions ('XQuery Injection')
High	XSLT Injection	OPT.CSHARP.XSLTInjection	XSLTInjection: Avoid using non-neutralized user-controlled input when creating XSL stylesheets
High	Review Visible Event Handlers	OPT.CSHARP.Csharp.ReviewVisibleEventHandlers	ReviewVisibleEventHandlers: A public or protected event-handling method was detected
High	Information Exposure Through Error Message	OPT.CSHARP.SEC.InformationExposureThroughErrorMessage	InformationExposureThroughErrorMessage: Avoid sensitive information exposure through error messages
High	Insecure Email Transport	OPT.CSHARP.SEC.InsecureEmailTransport	InsecureEmailTransport: Insecure Mail Transport
High	Insecure Randomness	OPT.CSHARP.InsecureRandomness	InsecureRandomness: Standard pseudo-random number generators cannot withstand cryptographic attacks
High	Hardcoded Crypto Key	OPT.CSHARP.SEC.HardcodedCryptoKey	HardcodedCryptoKey: Use of Hard-coded Cryptographic Key
High	Hardcoded Salt	OPT.CSHARP.SEC.HardcodedSalt	HardcodedSalt: A hardcoded salt can compromise system security

Severity	Contrast rule	Engine rule ID	Description
High	Insecure Transport	OPT.CSHARP.SEC.InsecureTransport	InsecureTransport: Insecure transport
High	Proper Padding With Public Key Crypto	OPT.CSHARP.SEC.ProperPaddingWithPublicKeyCrypto	ProperPaddingWithPublicKeyCrypto: Use of RSA Algorithm without Optimal Asymmetric Encryption Padding (OAEP)
High	Server Insecure Transport	OPT.CSHARP.SEC.ServerInsecureTransport	ServerInsecureTransport: Insecure transport [HTTP servers]
High	Weak Encryption	OPT.CSHARP.WeakEncryption	WeakEncryption: Insufficient RSA key length
Info	Avoid Unneeded Calls On String	OPT.CSHARP.Csharp.AvoidUnneededCallsOnString	AvoidUnneededCallsOnString: Avoid unnecessary calls on string objects
Info	Identifiers Should Have Correct Suffix	OPT.CSHARP.Csharp.IdentifiersShouldHaveCorrectSuffix	IdentifiersShouldHaveCorrectSuffix: An identifier does not have the correct suffix
Info	Identifiers Should Not Contain Underscores	OPT.CSHARP.Csharp.IdentifiersShouldNotContainUnderscores	IdentifiersShouldNotContainUnderscores: An identifier contains the underscore character
Info	Identifiers Should Not Have Incorrect Suffix	OPT.CSHARP.Csharp.IdentifiersShouldNotHaveIncorrectSuffix	IdentifiersShouldNotHaveIncorrectSuffix: An identifier has an incorrect suffix
Info	Normalize Strings To Uppercase	OPT.CSHARP.Csharp.NormalizeStringsToUppercase	NormalizeStringsToUppercase: Do not convert strings to lower-case
Info	Parameter Names Should Not Match Member Names	OPT.CSHARP.Csharp.ParameterNamesShouldNotMatchMemberNames	ParameterNamesShouldNotMatchMemberNames: Parameter names should not match member names
Info	Property Type	OPT.CSHARP.Csharp.PropertyType	PropertyType: A property must have a name similar to its type
Info	Use Exception Constructor	OPT.CSHARP.Csharp.UseExceptionConstructor	UseExceptionConstructor: Illegal exception throw: Exceptions must be created in separate methods
Info	Do Not Initialize Unnecessarily	OPT.CSHARP.DoNotInitializeUnnecessarily	DoNotInitializeUnnecessarily: Do not initialize variables unnecessarily
Info	Use Literals Where Appropriate	OPT.CSHARP.UseLiteralsWhereAppropriate	UseLiteralsWhereAppropriate: Do not declare static and readonly fields, that are initialized with a value
Low	Avoid Language Specific Type Names In Parameters	OPT.CSHARP.AvoidLanguageSpecificTypeNamesInParameters	AvoidLanguageSpecificTypeNamesInParameters: Avoid names of parameters that contain language-specific type name, in public methods
Low	Avoid System Output Stream	OPT.CSHARP.AvoidSystemOutputStream	AvoidSystemOutputStream: Using Console.WriteLine or Console.Error rather than a dedicated logging interface, makes it more difficult to monitor behavior of the software
Low	Avoid Type Names In Parameters	OPT.CSHARP.AvoidTypeNamesInParameters	AvoidTypeNamesInParameters: Avoid names of parameters that contain type's names, in public methods
Low	Avoid Unnecessary String Creation	OPT.CSHARP.AvoidUnnecessaryStringCreation	AvoidUnnecessaryStringCreation: Avoid creating unnecessary strings through a call to System.String.ToLower or System.String.ToUpper

Severity	Contrast rule	Engine rule ID	Description
Low	Collection Properties Should Be Read Only	OPT.CSHARP.CollectionPropertiesShouldBeReadOnly	CollectionPropertiesShouldBeReadOnly: Collection properties should be read-only
Low	Consider Converting Method To Property	OPT.CSHARP.ConsiderConvertingMethodToProperty	ConsiderConvertingMethodToProperty: Consider converting getter/setter methods to properties
Low	Attribute String Literals Should Parse Correctly	OPT.CSHARP.Csharp.AttributeStringLiteralsShouldParseCorrectly	AttributeStringLiteralsShouldParseCorrectly: Attribute's literal value should be correctly written
Low	Avoid Exceptions In Implicit Transformation	OPT.CSHARP.Csharp.AvoidExceptionsInImplicitTransformation	AvoidExceptionsInImplicitTransformation: Avoid throwing exceptions in implicit operators
Low	Avoid Long Methods	OPT.CSHARP.Csharp.AvoidLongMethods	AvoidLongMethods: Do not encode large methods
Low	Avoid Long Parameter Lists	OPT.CSHARP.Csharp.AvoidLongParameterLists	AvoidLongParameterLists: Do not encode methods with too many parameters
Low	Avoid Namespaces With Few Types	OPT.CSHARP.Csharp.AvoidNamespacesWithFewTypes	AvoidNamespacesWithFewTypes: Avoid namespaces with few types
Low	Avoid Non Stored Procedure Commands	OPT.CSHARP.Csharp.AvoidNonStoredProcedureCommands	AvoidNonStoredProcedureCommands: Avoid using non stored-procedure database operations
Low	Avoid Type Get Type For Constant Strings	OPT.CSHARP.Csharp.AvoidTypeGetTypeForConstantStrings	AvoidTypeGetTypeForConstantStrings: Do not call Type.GetType() with constant string values
Low	Call Base Class Methods On I Serializable Types	OPT.CSHARP.Csharp.CallBaseClassMethodsOnISerializableTypes	CallBaseClassMethodsOnISerializableTypes: Call base class methods on ISerializable types
Low	Check New Exception Without Throwing	OPT.CSHARP.Csharp.CheckNewExceptionWithoutThrowing	CheckNewExceptionWithoutThrowing: Exception instantiation not used
Low	Consider Custom Accessors For Non Visible Events Rule	OPT.CSHARP.Csharp.ConsiderCustomAccessorsForNonVisibleEventsRule	ConsiderCustomAccessorsForNonVisibleEventsRule: For non visible events, evaluate using custom accessors instead of default ones
Low	Delegates Passed To Native Code Must Include Exception Handling Rule	OPT.CSHARP.Csharp.DelegatesPassedToNativeCodeMustIncludeExceptionHandlingRule	DelegatesPassedToNativeCodeMustIncludeExceptionHandlingRule: Enclose with a catch handler entire block for delegates passed to native code
Low	Do Not Cast Unnecessarily	OPT.CSHARP.Csharp.DoNotCastUnnecessarily	DoNotCastUnnecessarily: A method performs duplicate casts on one of its arguments or variables
Low	Do Not Destroy Stack Trace Rule	OPT.CSHARP.Csharp.DoNotDestroyStackTraceRule	DoNotDestroyStackTraceRule: Catch handlers should rethrow original exception instead of throwing the same exception
Low	Do Not Hardcode Locale Specific Strings	OPT.CSHARP.Csharp.DoNotHardcodeLocaleSpecificStrings	DoNotHardcodeLocaleSpecificStrings: Do not hardcode locale specific strings

Severity	Contrast rule	Engine rule ID	Description
Low	Do Not Indirectly Expose Methods With Link Demands	OPT.CSHARP.Csharp.DoNotIndirectlyExposeMethodsWithLinkDemands	DoNotIndirectlyExposeMethodsWithLinkDemands: Do not indirectly expose methods with link demands
Low	Do Not Pass Literals As Localized Parameters	OPT.CSHARP.Csharp.DoNotPassLiteralsAsLocalizedParameters	DoNotPassLiteralsAsLocalizedParameters: String literal passed as parameter should be localizable
Low	Do Not Raise Exceptions In Unexpected Locations	OPT.CSHARP.Csharp.DoNotRaiseExceptionsInUnexpectedLocations	DoNotRaiseExceptionsInUnexpectedLocations: Do not raise exceptions in unexpected locations
Low	Do Not Use Thread Static With Instance Fields	OPT.CSHARP.Csharp.DoNotUseThreadStaticWithInstanceFields	DoNotUseThreadStaticWithInstanceFields: Do not use 'ThreadStatic' with instance fields
Low	Overloading Equals Reference Types	OPT.CSHARP.Csharp.OverloadingEqualsReferenceTypes	OverloadingEqualsReferenceTypes: Do not overload equality operator on reference types
Low	Override Equals On Value Types	OPT.CSHARP.Csharp.OverrideEqualsOnValueTypes	OverrideEqualsOnValueTypes: A public virtual method does not override Equals
Low	Properties Should Not Return Arrays	OPT.CSHARP.Csharp.PropertiesShouldNotReturnArrays	PropertiesShouldNotReturnArrays: Properties should not return arrays
Low	Property Names Should Not Match Get Methods	OPT.CSHARP.Csharp.PropertyNamesShouldNotMatchGetMethods	PropertyNamesShouldNotMatchGetMethods: Property names should not match get methods
Low	Review Unused Parameters	OPT.CSHARP.Csharp.ReviewUnusedParameters	ReviewUnusedParameters: Method parameters not used
Low	Specify Message Box Options	OPT.CSHARP.Csharp.SpecifyMessageBoxOptions	SpecifyMessageBoxOptions: Specify MessageBoxOptions
Low	Test For Empty Strings Using Length	OPT.CSHARP.Csharp.TestForEmptyStringsUsingLength	TestForEmptyStringsUsingLength: Compare with empty string using 'Equals'
Low	Type Names Should Not Match Namespaces	OPT.CSHARP.Csharp.TypeNamesShouldNotMatchNamespaces	TypeNamesShouldNotMatchNamespaces: Type names should not match namespaces
Low	Types That Own Native Resources Should Be Disposable	OPT.CSHARP.Csharp.TypesThatOwnNativeResourcesShouldBeDisposable	TypesThatOwnNativeResourcesShouldBeDisposable: Types that own native resources should be disposable
Low	Use Constructors To Initialize Properties	OPT.CSHARP.Csharp.UseConstructorsToInitializeProperties	UseConstructorsToInitializeProperties: Initialize properties by using constructors
Low	Use Constructors To Set Properties	OPT.CSHARP.Csharp.UseConstructorsToSetProperties	UseConstructorsToSetProperties: Set properties by using constructors with parameters
Low	Write Static Field From Instance Method	OPT.CSHARP.Csharp.WriteStaticFieldFromInstanceMethod	WriteStaticFieldFromInstanceMethod: Do not write in static fields from instance methods
Low	Disposable Types Should Declare Finalizer	OPT.CSHARP.DisposableTypesShouldDeclareFinalizer	DisposableTypesShouldDeclareFinalizer: A type that implements System.IDisposable and has unmanaged fields of unmanaged types, should have a finalizer

Severity	Contrast rule	Engine rule ID	Description
Low	Do Not Assign Params Not Out Or Ref	OPT.CSHARP.DoNotAssignParamsNotOutOrRef	DoNotAssignParamsNotOutOrRef: Do not assign method parameters which are not marked as 'out' or 'ref'
Low	Do Not Concatenate Strings Inside Loops	OPT.CSHARP.DoNotConcatenateStringsInsideLoops	DoNotConcatenateStringsInsideLoops: Do not concatenate string values inside loops
Low	Do Not Mark Enums With Flags	OPT.CSHARP.DoNotMarkEnumsWithFlags	DoNotMarkEnumsWithFlags: The values of an enumeration that has System.FlagsAttribute attribute, should be powers of two
Low	Do Not Prefix Enum Values With Type Name	OPT.CSHARP.DoNotPrefixEnumValuesWithTypeName	DoNotPrefixEnumValuesWithTypeName: Names of enumeration's members should not start with the enumeration's name
Low	Identifiers Should Be Cased Correctly	OPT.CSHARP.IdentifiersShouldBeCasedCorrectly	IdentifiersShouldBeCasedCorrectly: Use correctly "pascal-case" and "camel-case" according to the agreement established
Low	Implement Serialization Constructors	OPT.CSHARP.ImplementSerializationConstructors	ImplementSerializationConstructors: A type that implements ISerializable, should implement a serialization constructor
Low	Implement Serialization Methods Correctly	OPT.CSHARP.ImplementSerializationMethodsCorrectly	ImplementSerializationMethodsCorrectly: Implement a correct signature for methods that handle serialization events
Low	Initialize Value Type Static Fields Inline	OPT.CSHARP.InitializeValueTypeStaticFieldsInline	InitializeValueTypeStaticFieldsInline: Avoid explicitly declare a static constructor
Low	Instantiate Argument Exceptions Correctly	OPT.CSHARP.InstantiateArgumentExceptionsCorrectly	InstantiateArgumentExceptionsCorrectly: Avoid calling default constructor of ArgumentException
Low	Operator Overloads Have Named Alternates	OPT.CSHARP.OperatorOverloadsHaveNamedAlternates	OperatorOverloadsHaveNamedAlternates: When a type overrides an operator, it is recommended to also override the alternate method
Low	Overload Operator Equals On Overriding Equals	OPT.CSHARP.OverloadOperatorEqualsOnOverridingEquals	OverloadOperatorEqualsOnOverridingEquals: When a type overrides System.Object.Equals method, it also should override the operator
Low	Prefer Jagged Arrays Over Multidimensional	OPT.CSHARP.PreferJaggedArraysOverMultidimensional	PreferJaggedArraysOverMultidimensional: Do not declare multidimensional array
Low	Use Managed Equivalents Of Win32 Api	OPT.CSHARP.UseManagedEquivalentsOfWin32Api	UseManagedEquivalentsOfWin32Api: Use managed equivalents of Win32 API
Low	Constants Should Be Transparent	OPT.CSHARP.ConstantsShouldBeTransparent	ConstantsShouldBeTransparent: A field constant or an enumeration member should be transparent
Low	Information Exposure Through Debug Log	OPT.CSHARP.SEC.InformationExposureThroughDebugLog	InformationExposureThroughDebugLog: Avoid exposing sensible information through log
Medium	Unsafe Cookie Rule	OPT.CSHARP.SEC.UnsafeCookieRule	UnsafeCookieRule: Generate server-side cookies with adequate security properties
Medium	Avoid Null Reference Exception	OPT.CSHARP.AvoidNullReferenceException	AvoidNullReferenceException: Use of NullPointerException Catch to Detect Null Pointer Dereference
Medium	Avoid Readonly Mutable Types	OPT.CSHARP.AvoidReadonlyMutableTypes	AvoidReadonlyMutableTypes: Do not declare externally visible read-only fields with mutable types

Severity	Contrast rule	Engine rule ID	Description
Medium	Call GC Keep Alive When Using a Native Resource	OPT.CSHARP.CallGCKeepAliveWhenUsingaNativeResources	CallGCKeepAliveWhenUsingaNativeResource: GC.KeepAlive should be called in methods that use unmanaged resources
Medium	Critical Types Must Not Participate In Type Equivalence	OPT.CSHARP.CriticalTypesMustNotParticipateInTypeEquivalence	CriticalTypesMustNotParticipateInTypeEquivalence: SecurityCriticalAttribute should not be used in members, or types that participate in type equivalence
Medium	Array Fields Should Not Be Read Only	OPT.CSHARP.Csharp.ArrayFieldsShouldNotBeReadOnly	ArrayFieldsShouldNotBeReadOnly: Array fields should not be read only
Medium	Attribute Class Suffix	OPT.CSHARP.Csharp.AttributeClassSuffix	AttributeClassSuffix: Attribute classes names must contain the 'Attribute' suffix
Medium	Avoid Calling Problematic Methods	OPT.CSHARP.Csharp.AvoidCallingProblematicMethods	AvoidCallingProblematicMethods: Potentially dangerous call
Medium	Avoid Custom Application Exceptions	OPT.CSHARP.Csharp.AvoidCustomApplicationExceptions	AvoidCustomApplicationExceptions: Classes that extend ApplicationException should not be used
Medium	Avoid Empty Constructors In Structs	OPT.CSHARP.Csharp.AvoidEmptyConstructorsInStructs	AvoidEmptyConstructorsInStructs: Avoid empty constructors in structures
Medium	Avoid Indexers In Non Collection Classes	OPT.CSHARP.Csharp.AvoidIndexersInNonCollectionClasses	AvoidIndexersInNonCollectionClasses: Avoid indexers in non-collections classes
Medium	Avoid Large Methods	OPT.CSHARP.Csharp.AvoidLargeMethods	AvoidLargeMethods: Avoid functions and methods with too many lines of code
Medium	Avoid Large Structure	OPT.CSHARP.Csharp.AvoidLargeStructure	AvoidLargeStructure: Avoid creating too-large structures
Medium	Avoid Uninstantiated Internal Classes	OPT.CSHARP.Csharp.AvoidUninstantiatedInternalClasses	AvoidUninstantiatedInternalClasses: Avoid uninstantiated internal classes
Medium	Avoid Unsealed Concrete Attributes Rule	OPT.CSHARP.Csharp.AvoidUnsealedConcreteAttributesRule	AvoidUnsealedConcreteAttributesRule: Avoid attributes defined as unsealed but not abstract
Medium	Call Get Last Error Immediately After P Invoke	OPT.CSHARP.Csharp.CallGetLastErrorImmediatelyAfterPInvoke	CallGetLastErrorImmediatelyAfterPInvoke: GetLastError immediately after P/Invoke
Medium	Check New Thread Without Start	OPT.CSHARP.Csharp.CheckNewThreadWithoutStart	CheckNewThreadWithoutStart: Avoid creating unstarted threads
Medium	Clone Method Should Not Return Null	OPT.CSHARP.Csharp.CloneMethodShouldNotReturnNull	CloneMethodShouldNotReturnNull: An overridden Clone() method should never return null
Medium	Collection Class Suffix	OPT.CSHARP.Csharp.CollectionClassSuffix	CollectionClassSuffix: Collections classes names must end with the 'Collection' suffix
Medium	Collections Should Implement Generic Interface	OPT.CSHARP.Csharp.CollectionsShouldImplementGenericInterface	CollectionsShouldImplementGenericInterface: Collections should implement the generic interface
Medium	Com Visible Type Base Types Should Be Com Visible	OPT.CSHARP.Csharp.ComVisibleTypeBaseTypesShouldBeComVisible	ComVisibleTypeBaseTypesShouldBeComVisible: COM visible type derive from a non COM visible type

Severity	Contrast rule	Engine rule ID	Description
Medium	Consider Passing Base Types As Parameters	OPT.CSHARP.Csharp.ConsiderPassingBaseTypesAsParameters	ConsiderPassingBaseTypesAsParameters: Consider passing base types as parameters
Medium	Declare Types In Namespaces	OPT.CSHARP.Csharp.DeclareTypesInNamespaces	DeclareTypesInNamespaces: Declare type namespaces
Medium	Default Parameters Should Not Be Used	OPT.CSHARP.Csharp.DefaultParametersShouldNotBeUsed	DefaultParametersShouldNotBeUsed: Do not use default parameters
Medium	Disable Debugging Code Rule	OPT.CSHARP.Csharp.DisableDebuggingCodeRule	DisableDebuggingCodeRule: Avoid using Console.WriteLine
Medium	Disposable Fields Should Be Disposed	OPT.CSHARP.Csharp.DisposableFieldsShouldBeDisposed	DisposableFieldsShouldBeDisposed: Call Dispose method of fields that implements System.IDisposable
Medium	Do Not Call Overridable Methods In Constructors	OPT.CSHARP.Csharp.DoNotCallOverridableMethodsInConstructors	DoNotCallOverridableMethodsInConstructors: Virtual method call from constructor
Medium	Do Not Catch General Exception Types	OPT.CSHARP.Csharp.DoNotCatchGeneralExceptionTypes	DoNotCatchGeneralExceptionTypes: Declaration of Catch for Generic Exception Types
Medium	Do Not Declare Virtual Members In Sealed Types	OPT.CSHARP.Csharp.DoNotDeclareVirtualMembersInSealedTypes	DoNotDeclareVirtualMembersInSealedTypes: Do not declare virtual and not final members in sealed classes
Medium	Do Not Decrease Inherited Member Visibility	OPT.CSHARP.Csharp.DoNotDecreaseInheritedMemberVisibility	DoNotDecreaseInheritedMemberVisibility: Do not decrease inherited member visibility
Medium	Do Not Ignore Method Results	OPT.CSHARP.Csharp.DoNotIgnoreMethodResults	DoNotIgnoreMethodResults: Do not ignore returning value of methods
Medium	Do Not Nest Generic Types In Member Signatures	OPT.CSHARP.Csharp.DoNotNestGenericTypesInMemberSignatures	DoNotNestGenericTypesInMemberSignatures: Do not nest generic types in externally visible member signatures
Medium	Do Not Raise Exceptions In Exception Clauses	OPT.CSHARP.Csharp.DoNotRaiseExceptionsInExceptionClauses	DoNotRaiseExceptionsInExceptionClauses: Do not raise exceptions in exception clauses
Medium	Do Not Raise Reserved Exception Types	OPT.CSHARP.Csharp.DoNotRaiseReservedExceptionTypes	DoNotRaiseReservedExceptionTypes: Do not raise reserved exception types
Medium	Do Not Use Timers That Prevent Power State Changes	OPT.CSHARP.Csharp.DoNotUseTimersThatPreventPowerStateChanges	DoNotUseTimersThatPreventPowerStateChanges: Avoid timers that prevent power state changes
Medium	Double Check Locking Rule	OPT.CSHARP.Csharp.DoubleCheckLockingRule	DoubleCheckLockingRule: Incorrect usage of double checking when implementing Single-Checked Locking pattern
Medium	Equality Operator If Plus Or Minus	OPT.CSHARP.Csharp.EqualityOperatorIfPlusOrMinus	EqualityOperatorIfPlusOrMinus: Override equality operator if addition and subtraction operators are overridden
Medium	Exception Class Suffix	OPT.CSHARP.Csharp.ExceptionClassSuffix	ExceptionClassSuffix: Exception classes must end with the 'Exception' suffix
Medium	Exception Constructors	OPT.CSHARP.Csharp.ExceptionConstructors	ExceptionConstructors: Classes that extend Exception must implement all standard constructors

Severity	Contrast rule	Engine rule ID	Description
Medium	Implement IDisposable Correctly	OPT.CSHARP.Csharp.ImplementIDisposableCorrectly	ImplementIDisposableCorrectly: Implement IDisposable correctly
Medium	Implement IDisposable With Finalize	OPT.CSHARP.Csharp.ImplementIDisposableWithFinalize	ImplementIDisposableWithFinalize: Implement Dispose method provided by IDisposable interface
Medium	Implement ISerializable Correctly	OPT.CSHARP.Csharp.ImplementISerializableCorrectly	ImplementISerializableCorrectly: Implement ISerializable correctly
Medium	Implement Standard Exception Constructors	OPT.CSHARP.Csharp.ImplementStandardExceptionConstructors	ImplementStandardExceptionConstructors: Implement standard exception constructors
Medium	Interface Name	OPT.CSHARP.Csharp.InterfaceName	InterfaceName: Interface name format convention
Medium	Level2 Assemblies Should Not Contain Linkdemands	OPT.CSHARP.Csharp.Level2AssembliesShouldNotContainLinkdemands	Level2AssembliesShouldNotContainLinkdemands: A class or class member is using a LinkDemand in an application that is using Level 2 security
Medium	Mark Members As Static	OPT.CSHARP.Csharp.MarkMembersAsStatic	MarkMembersAsStatic: A method that only accesses class members should be marked 'static'
Medium	Members Should Not Expose Certain Concrete Types	OPT.CSHARP.Csharp.MembersShouldNotExposeCertainConcreteTypes	MembersShouldNotExposeCertainConcreteTypes: Members should not expose certain concrete types
Medium	Method Case	OPT.CSHARP.Csharp.MethodCase	MethodCase: Method names differ only in case within the same class
Medium	Namespace Case	OPT.CSHARP.Csharp.NamespaceCase	NamespaceCase: Namespace names differ only in case
Medium	Naming Class Namespace	OPT.CSHARP.Csharp.NamingClassNamespace	NamingClassNamespace: Namespace name cannot match the name of one of its containing classes
Medium	Nested Types Should Not Be Visible	OPT.CSHARP.Csharp.NestedTypesShouldNotBeVisible	NestedTypesShouldNotBeVisible: An externally visible type contains an externally visible type declaration
Medium	Num Max Class By Namespace	OPT.CSHARP.Csharp.NumMaxClassByNamespace	NumMaxClassByNamespace: Avoid an excessive number of classes per package/namespace
Medium	Only Flags Enums Should Have Plural Names	OPT.CSHARP.Csharp.OnlyFlagsEnumsShouldHavePluralNames	OnlyFlagsEnumsShouldHavePluralNames: Externally visible enumeration ends in a plural word and is not marked with the Flags attribute
Medium	Operations Should Not Overflow	OPT.CSHARP.Csharp.OperationsShouldNotOverflow	OperationsShouldNotOverflow: Operations should not overflow
Medium	Parameter Case	OPT.CSHARP.Csharp.ParameterCase	ParameterCase: Parameter names differ only in case in a method declaration
Medium	Pass System Obj Instead Of String	OPT.CSHARP.Csharp.PassSystemObjInsteadOfString	PassSystemObjInsteadOfString: Pass System.Uri objects instead of strings
Medium	Pointers Should Not Be Visible	OPT.CSHARP.Csharp.PointersShouldNotBeVisible	PointersShouldNotBeVisible: Pointers should not be visible
Medium	Properties Should Not Be Write Only	OPT.CSHARP.Csharp.PropertiesShouldNotBeWriteOnly	PropertiesShouldNotBeWriteOnly: Avoid write-only properties
Medium	Property Case	OPT.CSHARP.Csharp.PropertyCase	PropertyCase: Property names within the same class differ only in case
Medium	Rethrow To Preserve Stack Details	OPT.CSHARP.Csharp.RethrowToPreserveStackDetails	RethrowToPreserveStackDetails: Do not rethrow exceptions explicitly

Severity	Contrast rule	Engine rule ID	Description
Medium	Set Locale For Data Types	OPT.CSHARP.Csharp.SetLocaleForDataTypes	SetLocaleForDataTypes: Set locale proper data types
Medium	Specify Culture Info	OPT.CSHARP.Csharp.SpecifyCultureInfo	SpecifyCultureInfo: Specify CultureInfo
Medium	Specify String Comparison	OPT.CSHARP.Csharp.SpecifyStringComparison	SpecifyStringComparison: Specify StringComparison
Medium	Static Holder Types Should Be Sealed	OPT.CSHARP.Csharp.StaticHolderTypesShouldBeSealed	StaticHolderTypesShouldBeSealed: Class containing only static members should be declared sealed
Medium	Static Holder Types Should Not Have Constructors	OPT.CSHARP.Csharp.StaticHolderTypesShouldNotHaveConstructors	StaticHolderTypesShouldNotHaveConstructors: Static holder types should not have constructors
Medium	Type Case	OPT.CSHARP.Csharp.TypeCase	TypeCase: Type names differ only in case
Medium	Types Should Not Extend Certain Base Types	OPT.CSHARP.Csharp.TypesShouldNotExtendCertainBaseTypes	TypesShouldNotExtendCertainBaseTypes: Types should not extend certain base types
Medium	Uri Parameters Should Not Be Strings	OPT.CSHARP.Csharp.UriParametersShouldNotBeStrings	UriParametersShouldNotBeStrings: URI parameters should not be strings
Medium	Uri Return Values Should Not Be Strings	OPT.CSHARP.Csharp.UriReturnValuesShouldNotBeStrings	UriReturnValuesShouldNotBeStrings: The return value of a method contains 'uri', 'Uri', 'urn', 'Urn', or 'Url', and returns a string
Medium	Use Generic Event Handler Instances	OPT.CSHARP.Csharp.UseGenericEventHandlerInstances	UseGenericEventHandlerInstances: Use generic event handler instances
Medium	Use Safe Handle To Encapsulate Native Resources	OPT.CSHARP.Csharp.UseSafeHandleToEncapsulateNativeResources	UseSafeHandleToEncapsulateNativeResources: Use of System.IntPtr
Medium	Validate Arguments Of Public Methods	OPT.CSHARP.Csharp.ValidateArgumentsOfPublicMethods	ValidateArgumentsOfPublicMethods: Check parameters of externally visible methods
Medium	Warn Of Assignations In Conditional Statements	OPT.CSHARP.Csharp.WarnOfAssignationsInConditionalStatements	WarnOfAssignationsInConditionalStatements: Assignment expression in an if condition
Medium	Classes Are Strongly Internally Coupled	OPT.CSHARP.ClassesAreStronglyInternallyCoupled	ClassesAreStronglyInternallyCoupled: Classes that are internally strongly coupled must be avoided
Medium	Dispose Methods Should Call Suppress Finalize	OPT.CSHARP.DisposeMethodsShouldCallSuppressFinalize	DisposeMethodsShouldCallSuppressFinalize: The Dispose method of a class that implements System.IDisposable should call GC.SuppressFinalize
Medium	Method Security Should Be Superset Of Type	OPT.CSHARP.MethodSecurityShouldBeSupersetOfType	MethodSecurityShouldBeSupersetOfType: Security of methods should be a subset of security of types
Medium	MVC Post In Controllers	OPT.CSHARP.MVCPostInControllers	MVCPostInControllers: Restrict allowed HTTP verbs for state-change operations in MVC controllers
Medium	Potential Infinite Loop	OPT.CSHARP.PotentialInfiniteLoop	PotentialInfiniteLoop: Loop with Unreachable Exit Condition ('Infinite Loop')
Medium	Provide Correct Arguments To Formatting Methods	OPT.CSHARP.ProvideCorrectArgumentsToFormattingMethods	ProvideCorrectArgumentsToFormattingMethods: The format argument passed to System.String.Format does not match with objects passed as parameters

Severity	Contrast rule	Engine rule ID	Description
Medium	Provide Deserialization Methods For Optional Fields	OPT.CSHARP.ProvideDeserializationMethodsForOptionalFields	ProvideDeserializationMethodsForOptionalFields: Provide methods for the de-serialization of fields marked with OptionalFieldAttribute
Medium	Review Declarative Security On Value Types	OPT.CSHARP.ReviewDeclarativeSecurityOnValueTypes	ReviewDeclarativeSecurityOnValueTypes: Avoid using declarative security in value types
Medium	Review Imperative Security	OPT.CSHARP.ReviewImperativeSecurity	ReviewImperativeSecurity: Avoid using imperative security whenever possible
Medium	Http Request Value Shadowing	OPT.CSHARP.SEC.HttpRequestValueShadowing	HttpRequestValueShadowing: Request data is accessed in an ambiguous way, which can leave it open to attack
Medium	Main Method In Web Application	OPT.CSHARP.SEC.MainMethodInWebApplication	MainMethodInWebApplication: Main() method is not allowed in web application
Medium	Avoid Host Name Checks	OPT.CSHARP.SEC.AvoidHostNameChecks	AvoidHostNameChecks: Avoid checks on host side hostname, that are not reliable due to DNS poisoning
Medium	System Information Leak	OPT.CSHARP.SystemInformationLeak	SystemInformationLeak: Exposure of System Data to an Unauthorized Control Sphere
Medium	Transparent Methods Must Not Call Native Code	OPT.CSHARP.TransparentMethodsMustNotCallNativeCode	TransparentMethodsMustNotCallNativeCode: A transparent method should not make calls to native code
Medium	Transparent Methods Must Not Handle Process Corrupting Exceptions	OPT.CSHARP.TransparentMethodsMustNotHandleProcessCorruptingExceptions	TransparentMethodsMustNotHandleProcessCorruptingExceptions: A transparent method must not have HandleProcessCorruptedStateExceptions attribute
Medium	Transparent Methods Should Not Be Protected With Link Demands	OPT.CSHARP.TransparentMethodsShouldNotBeProtectedWithLinkDemands	TransparentMethodsShouldNotBeProtectedWithLinkDemands: A transparent method should not require LinkDemand
Medium	Transparent Methods Should Not Demand	OPT.CSHARP.TransparentMethodsShouldNotDemand	TransparentMethodsShouldNotDemand: A transparent method should not require SecurityAction.Demand, and should not call CodeAccessPermission.Demand method
Medium	Transparent Methods Should Not Load Assemblies From Byte Arrays	OPT.CSHARP.TransparentMethodsShouldNotLoadAssembliesFromByteArrays	TransparentMethodsShouldNotLoadAssembliesFromByteArrays: A transparent method should not load an assembly from a byte array using the Assembly.Load method
Medium	Type Link Demands Require Inheritance Demands	OPT.CSHARP.TypeLinkDemandsRequireInheritanceDemands	TypeLinkDemandsRequireInheritanceDemands: A public type protected with link demand requires inheritance demand
Medium	Unchecked Return Value	OPT.CSHARP.UncheckedReturnValue	UncheckedReturnValue: Unchecked return value.
Medium	Unchecked Input In Loop Condition	OPT.CSHARP.UncheckedInputInLoopCondition	UncheckedInputInLoopCondition: Unchecked input in loop condition
Medium	Unused Private Method	OPT.CSHARP.UnusedPrivateMethod	UnusedPrivateMethod: Avoid unused private methods and constructors
Medium	Do Not Expose Fields In Secured Type	OPT.CSHARP.Csharp.DoNotExposeFieldsInSecuredType	DoNotExposeFieldsInSecuredType: Do not declare public types that are secured but also expose its fields

Severity	Contrast rule	Engine rule ID	Description
Medium	Review Suppress Unmanaged Code Security Usage	OPT.CSHARP.Csharp.ReviewSuppressUnmanagedCodeSecurityUsage	ReviewSuppressUnmanagedCodeSecurityUsage: Do not use the 'SuppressUnmanagedCodeSecurity' attribute
Medium	MVC Remove Version Header	OPT.CSHARP.MVCRemoveVersionHeader	MVCRemoveVersionHeader: Remove ASP.NET MVC version from HTTP headers
Medium	P Invokes Should Not Be Safe Critical	OPT.CSHARP.PInvokesShouldNotBeSafeCritical	PInvokesShouldNotBeSafeCritical: A P/Invoke declaration should not have the SecuritySafeCritical attribute
Medium	Hardcoded Credential	OPT.CSHARP.SEC.HardcodedCredential	HardcodedCredential: Use of Hard-coded Credentials
Medium	Hardcoded Network Address	OPT.CSHARP.SEC.HardcodedNetworkAddress	HardcodedNetworkAddress: Network address should not be hardcoded
Medium	Plaintext Storage Of Password	OPT.CSHARP.SEC.PlaintextStorageOfPassword	PlaintextStorageOfPassword: Plaintext Storage of a Password
Medium	Serializable Class Containing Sensitive Data	OPT.CSHARP.SEC.SerializableClassContainingSensitiveData	SerializableClassContainingSensitiveData: Serializable Class Containing Sensitive Data
Medium	Secured Types Should Not Expose Fields	OPT.CSHARP.SecuredTypesShouldNotExposeFields	SecuredTypesShouldNotExposeFields: Types secured with Link Demands should not expose fields
Medium	Secure Serialization Constructors	OPT.CSHARP.SecureSerializationConstructors	SecureSerializationConstructors: Serialized constructors should be protected with security demands
Medium	Transparency Annotations Should Not Conflict	OPT.CSHARP.TransparencyAnnotationsShouldNotConflict	TransparencyAnnotationsShouldNotConflict: The security attribute of a type should have the same transparency that the security attribute of the members that it contains

COBOL Scan rules

Contrast Scan supports these rules for COBOL.

Severity	Contrast rule	Engine rule ID	Description
Critical	Avoid Access Not Indexed Table Big	OPT.COBOLE.AvoidAccessNotIndexedTableBig	AvoidAccessNotIndexedTableBig: Detected an access to a large table (no. of pages > 1000)
Critical	Avoid Access Without Index Big	OPT.COBOLE.AvoidAccessWithoutIndexBig	AvoidAccessWithoutIndexBig: Detected access where there is no index fields reported in the WHERE on a large table size (number of pages > 1000)
Critical	Avoid As In SQL Sentence	OPT.COBOLE.AvoidAsInSqlSentence	AvoidAsInSqlSentence: It is not allowed the use of statements that include temporary tables defined using the AS clause
Critical	Avoid Call Other Section Paragraphs	OPT.COBOLE.AvoidCallOtherSectionParagraphs	AvoidCallOtherSectionParagraphs: Call a paragraph from other section
Critical	Avoid Join With Cost Access	OPT.COBOLE.AvoidJoinWithCostAccess	AvoidJoinWithCostAccess: Not allowed to use JOIN containing costly accesses (R0, I0, MX) to one of the tables (no. of pages > 1000)

Severity	Contrast rule	Engine rule ID	Description
Critical	Avoid Paragraphs Out Of Sections	OPT.COBOL.AvoidParagraphsOutOfSections	AvoidParagraphsOutOfSections: Avoid paragraphs outside of sections
Critical	Check SQL Code After Sequence	OPT.COBOL.CheckSqlCodeAfterSequence	CheckSqlCodeAfterSequence: Check return codes (SQLCODE -359 and / or -845) in SEQUENCE objects with NEXT VALUE or PREVIOUS VALUE
Critical	Check SQLcode When Rowset	OPT.COBOL.CheckSqlcodeWhenRowset	CheckSqlcodeWhenRowset: Check rows and returned code (SQLCODE {}
Critical	Check Value Occur	OPT.COBOL.CheckValueOccur	CheckValueOccur: When use MULTIROW, the value of 'n' in FOR n ROWS option, has to be less or equal to OCCURS in the table that receives the rowset
Critical	Call Paragraph	OPT.COBOL.COBBP.CallParagraph	CallParagraph: Uncalled paragraph/ section
Critical	N D E S	OPT.COBOL.COD_COBOL.NDES	NDES: Do not divide PROCEDURE DIVISION into sections
Critical	N R	OPT.COBOL.COD_COBOL.NR	NR: Do not use RETURN inside EXEC CICS statements
Critical	Avoid Collisions In Procedure Names	OPT.COBOL.FIA_COBOL.AvoidCollisionsInProcedureNames	AvoidCollisionsInProcedureNames: Avoid duplicated section names, or paragraph names in same section
Critical	Close Open Files	OPT.COBOL.FIA_COBOL.CloseOpenFiles	CloseOpenFiles: Check that every opened file is closed
Critical	Close Open Input Output Files	OPT.COBOL.FIA_COBOL.CloseOpenInputOutputFiles	CloseOpenInputOutputFiles: Check that every opened (input or output) file is closed
Critical	Open Declared Files	OPT.COBOL.FIA_COBOL.OpenDeclaredFiles	OpenDeclaredFiles: Check that every declared file is opened
Critical	Read Or Write Open Files	OPT.COBOL.FIA_COBOL.ReadOrWriteOpenFiles	ReadOrWriteOpenFiles: Check that every opened file is read or written
Critical	Last Rows Invalid Checks	OPT.COBOL.LastRowsInvalidChecks	LastRowsInvalidChecks: Detected control of SQLCODE values ,ÄÄthat cannot be done.
Critical	Last Rows Valid Checks	OPT.COBOL.LastRowsValidChecks	LastRowsValidChecks: Missing control of SQLCODE values ,ÄÄneeded.
Critical	M L S	OPT.COBOL.MAN_COBOL.MLS	MLS: Avoid exceeding maximum number of lines per Cobol program
Critical	M S	OPT.COBOL.MAN_COBOL.MS	MS: Use a single program exit point (STOP or GOBACK) per program
Critical	D C	OPT.COBOL.RG_COBOL.DC	DC: With DIVIDE or COMPUTE with a division, add ON SIZE ERROR to control potential division by zero
Critical	F D S N	OPT.COBOL.RG_COBOL.FDSN	FDSN: Avoid FD without record descriptor
Critical	L R S	OPT.COBOL.RG_COBOL.LRS	LRS: Use LABEL RECORD IS STANDARD in file descriptors (FD)
Critical	N L F	OPT.COBOL.RG_COBOL.NLF	NLF: Avoid programmes with too many lines
Critical	N U R	OPT.COBOL.RG_COBOL.NUR	NUR: Do not use REPORT clause
Critical	Avoid Alter	OPT.COBOL.SEC.AvoidAlter	AvoidAlter: Avoid ALTER
Critical	Cobol Access Control DLI	OPT.COBOL.SEC.Cobol_AccessControlDLI	Cobol_AccessControlDLI: Check user input used in DL/I (IMS) queries

Severity	Contrast rule	Engine rule ID	Description
Critical	Cobol Access Control Database	OPT.COBOL.SEC.Cobol_AccessControlDatabase	Cobol_AccessControlDatabase: Authorization Bypass Through User-Controlled SQL Primary Key
Critical	Dynamic Storage Leak Rule	OPT.COBOL.SEC.DynamicStorageLeakRule	DynamicStorageLeakRule: Potential dynamic storage area leak
Critical	Illegal Values For Pointers	OPT.COBOL.SEC.IllegalValuesForPointers	IllegalValuesForPointers: Access of Uninitialized Pointer
Critical	Path Traversal	OPT.COBOL.SEC.PathTraversal	PathTraversal: Avoid non-neutralized user-controlled input to be part of a pathname (file or directory) used in I/O operations
Critical	Pointer Arithmetic	OPT.COBOL.SEC.PointerArithmetic	PointerArithmetic: Avoid pointer arithmetic in Cobol
Critical	Avoid Duplicated Queries	OPT.COBOL.SQL_COBOL.AvoidDuplicatedQueries	AvoidDuplicatedQueries: Avoid duplicated SQL statements
Critical	Cursor For Update Where Current	OPT.COBOL.SQL_COBOL.CursorForUpdateWhereCurrent	CursorForUpdateWhereCurrent: If a CURSOR is declared FOR UPDATE, DELETE and UPDATE must be used with the WHERE CURRENT specification
Critical	Detect Unaware Cross Joins	OPT.COBOL.SQL_COBOL.DetectUnawareCrossJoins	DetectUnawareCrossJoins: Do not make "unnoticed" cartesian products in queries
Critical	Dont Select Known Fields	OPT.COBOL.SQL_COBOL.DontSelectKnownFields	DontSelectKnownFields: SELECT queries never should get fields used in the WHERE specification with {}
Critical	Fetch And Declare Same Fields	OPT.COBOL.SQL_COBOL.FetchAndDeclareSameFields	FetchAndDeclareSameFields: The number of fields to retrieve specified in the DECLARE CURSOR statement must be the same as the number of fields specified in the FETCH statement
Critical	Avoid Correlated Sub Selects	OPT.COBOL.SQL_COBOL.AvoidCorrelatedSubSelects	AvoidCorrelatedSubSelects: Avoid nested SELECTs that use columns defined in outer SELECTs
Critical	Cobol Access Control MQ	OPT.COBOL.SEC.Cobol_AccessControlMQ	Cobol_AccessControlMQ: Do not allow user input to control fields of MQSeries descriptor
Critical	Cobol Process Control	OPT.COBOL.SEC.Cobol_ProcessControl	Cobol_ProcessControl: Avoid calling subprogram where its name could be controlled by user input
Critical	Cobol Resource Injection	OPT.COBOL.SEC.Cobol_ResourceInjection	Cobol_ResourceInjection: Improper Control of Resource Identifiers ('Resource Injection')
Critical	Cross Site Scripting	OPT.COBOL.SEC.CrossSiteScripting	CrossSiteScripting: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
Critical	OS Command Injection	OPT.COBOL.SEC.OSCommandInjection	OSCommandInjection: Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
Critical	SQL Injection	OPT.COBOL.SEC.SqlInjection	SqlInjection: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
Critical	Cobol Hardcoded Password	OPT.COBOL.SEC.Cobol_HardcodedPassword	Cobol_HardcodedPassword: Hardcoded passwords can compromise system security in a way that cannot be easily remedied

Severity	Contrast rule	Engine rule ID	Description
Critical	HTTP Header Manipulation	OPT.COBOL.SEC.HTTPHeaderManipulation	HTTPHeaderManipulation: Unvalidated data in HTTP response header
Critical	Check Crypto Return Code	OPT.COBOL.SEC.CheckCryptoReturnCode	CheckCryptoReturnCode: Validate return code for cryptographic operations
High	Avoid XML	OPT.COBOL.AvoidXML	AvoidXML: Do not use (read and parse) XML in a cobol program
High	Avoid XML Generate	OPT.COBOL.AvoidXMLGenerate	AvoidXMLGenerate: Do not generate XML in a cobol program
High	Link Xctl With Commarea Length	OPT.COBOL.CICS.LinkXctlWithCommareaLength	LinkXctlWithCommareaLength: Specify LENGTH with COMMAREA in CICS LINK/XCTL/RETURN commands
High	Use Cics Explicit Error Handling	OPT.COBOL.CICS.UseCicsExplicitErrorHandling	UseCicsExplicitErrorHandling: In CICS commands, use error handling with RESP / NOHANDE and test result code
High	I S E	OPT.COBOL.COBBP.ISE	ISE: Close each IF with END-IF
High	R I B	OPT.COBOL.COBBP.RIB	RIB: FD descriptor must specify BLOCK CONTAINS 0 RECORDS
High	C O B N O M Call Naming Convention	OPT.COBOL.COBNOM_CallNamingConvention	COBNOM_CallNamingConvention: CALL naming conventions
High	Check File Status After I O	OPT.COBOL.FIA_COBOL.CheckFileStatusAfterIO	CheckFileStatusAfterIO: FILE STATUS should be checked after I/O operations
High	No Stmt After Program Termination	OPT.COBOL.FIA_COBOL.NoStmtAfterProgramTermination	NoStmtAfterProgramTermination: There should be no statements following STOP RUN / GOBACK / EXIT PROGRAM
High	Use Field W S	OPT.COBOL.FIA_COBOL.UseFieldWS	UseFieldWS: Check that every field declared in WORKING-STORAGE is used
High	W O E V	OPT.COBOL.FIA_COBOL.WOEV	WOEV: Use WHEN OTHER in EVALUATE
High	C N P	OPT.COBOL.MAN_COBOL.CNP	CNP: Comment paragraphs and procedure sections
High	E I F A	OPT.COBOL.MAN_COBOL.EIFA	EIFA: Avoid nesting IF too deeply
High	I N V F	OPT.COBOL.MAN_COBOL.INVF	INVF: Avoid programs with high integration complexity
High	M N N	OPT.COBOL.MAN_COBOL.MNN	MNN: DISPLAY instruction with non-normalised message codes
High	N S T	OPT.COBOL.MAN_COBOL.NST	NST: Limit the number of executable sentences per Cobol program
High	P COM	OPT.COBOL.MAN_COBOL.PCOM	PCOM: Avoid programs with low code comments rate
High	P R C D	OPT.COBOL.MAN_COBOL.PRCD	PRCD: Avoid deeply nested flow-control sentences
High	R A C C	OPT.COBOL.MAN_COBOL.RACC	RACC: Avoid programs/routines with high cyclomatic complexity
High	S COM	OPT.COBOL.MAN_COBOL.SCOM	SCOM: Avoid sections with low comment percentage
High	Not Used Fields	OPT.COBOL.NotUsedFields	NotUsedFields: Unused fields retrieved in SELECT or FETCH statements
High	H I F I	OPT.COBOL.OYR_COBOL.HIFI	HIFI: Avoid high fan-in procedures
High	H I F O	OPT.COBOL.OYR_COBOL.HIFO	HIFO: Avoid high fan-out procedures

Severity	Contrast rule	Engine rule ID	Description
High	Read Followed By At End Or Invalid Key	OPT.COBOL.ReadFollowedByAtEndOrInvalidKey	ReadFollowedByAtEndOrInvalidKey: READ statement without by AT END or INVALID KEY.
High	F C T	OPT.COBOL.RG_COBOL.FCT	FCT: Programs must have at least as many counters as tables and files declared in program
High	F S R	OPT.COBOL.RG_COBOL.FSR	FSR: Define FD as LABEL RECORD STANDARD, 0-record blocks, and recording mode fixed or variable
High	G O T O	OPT.COBOL.RG_COBOL.GOTO	GOTO: Avoid GO TO statements in program logic
High	N T P	OPT.COBOL.RG_COBOL.NTP	NTP: Do not use PERFORM ... THRU
High	Call Parameter Mismatch	OPT.COBOL.SEC.CallParameterMismatch	CallParameterMismatch: Parameter mismatch in CALL
High	Avoid Declared Unopened Cursors	OPT.COBOL.SQL_COBOL.AvoidDeclaredUnopenedCursors	AvoidDeclaredUnopenedCursors: If a CURSOR is declared, it must be opened
High	Avoid Included Tables And Not Accessed	OPT.COBOL.SQL_COBOL.AvoidIncludedTablesAndNotAccessed	AvoidIncludedTablesAndNotAccessed: Avoid included table definitions, not accessed after in the programme body
High	Avoid Opened Unclosed Cursors	OPT.COBOL.SQL_COBOL.AvoidOpenedUnclosedCursors	AvoidOpenedUnclosedCursors: If a CURSOR is opened, it must be closed
High	Avoid Opened Unused Cursors	OPT.COBOL.SQL_COBOL.AvoidOpenedUnusedCursors	AvoidOpenedUnusedCursors: If a CURSOR is opened, it must be used
High	Check SQLcode Or Indicator Vars In Select	OPT.COBOL.SQL_COBOL.CheckSqlcodeOrIndicatorVarsInSelect	CheckSqlcodeOrIndicatorVarsInSelect: Check for NULL properly (use host variables in SQL sentence or check for SQLCODE {})
High	Control SQLcode After Exec SQL	OPT.COBOL.SQL_COBOL.ControlSqlcodeAfterExecSql	ControlSqlcodeAfterExecSql: SQLCODE value should be checked after each EXEC SQL statement
High	No Current Clause	OPT.COBOL.SQL_COBOL.NoCurrentClause	NoCurrentClause: SQL queries with CURRENT clause are heavy-weighted and must be used only when necessary
High	Optimize Varchar Moves	OPT.COBOL.SQL_COBOL.OptimizeVarcharMoves	OptimizeVarcharMoves: Control data size on VARCHAR columns
High	Avoid Union	OPT.COBOL.SQL_COBOL.AvoidUnion	AvoidUnion: Avoid selects with UNION
High	Use The As Keyword	OPT.COBOL.SQL_COBOL.UseTheAsKeyword	UseTheAsKeyword: Use AS keyword when establishing an alias to tables
High	Avoid Numeric References In By Clauses	OPT.COBOL.SQL_COBOL.AvoidNumericReferencesInByClauses	AvoidNumericReferencesInByClauses: Do not refer to column names with number indexes in * BY clauses
High	No Accept From Untrusted Source	OPT.COBOL.SEC.NoAcceptFromUntrustedSource	NoAcceptFromUntrustedSource: Do not ACCEPT data from untrusted sources
High	No Active Debug	OPT.COBOL.SEC.NoActiveDebug	NoActiveDebug: Information Exposure Through Debug Information

Severity	Contrast rule	Engine rule ID	Description
High	Weak Crypto Hash	OPT.COBOL.SEC.WeakCryptoHash	WeakCryptoHash: Weak cryptographic hashes cannot guarantee data integrity
Info	Access In Loop Not Used Index	OPT.COBOL.AccessInLoopNotUsedIndex	AccessInLoopNotUsedIndex: Access detected in a loop to a medium table (no. pages > {})
Info	Avoid Accept From Console	OPT.COBOL.AvoidAcceptFromConsole	AvoidAcceptFromConsole: Using ACCEPT FROM CONSOLE is not allowed
Info	Avoid Access Not Indexed Table Small	OPT.COBOL.AvoidAccessNotIndexedTableSmall	AvoidAccessNotIndexedTableSmall: Detected a small table access (no. pages < {})
Info	Avoid Copy Procedure Division	OPT.COBOL.AvoidCopyProcedureDivision	AvoidCopyProcedureDivision: Using COPY in PROCEDURE DIVISION is not allowed
Info	Avoid Distinct	OPT.COBOL.AvoidDistinct	AvoidDistinct: Avoid the DISTINCT operator
Info	Avoid Divide0	OPT.COBOL.AvoidDivide0	AvoidDivide0: Avoid division by 0
Info	Avoid If Numeric Alphabetic	OPT.COBOL.AvoidIfNumericAlphabetic	AvoidIfNumericAlphabetic: Avoid using IF NUMERIC and IF ALPHABETIC
Info	Avoid Include Procedure Division	OPT.COBOL.AvoidIncludeProcedureDivision	AvoidIncludeProcedureDivision: Using INCLUDE in PROCEDURE DIVISION is not allowed
Info	Avoid Mix SQL Code	OPT.COBOL.AvoidMixSqlCode	AvoidMixSqlCode: In a control statement does not combine program variables with return codes (SQL-CODE)
Info	Avoid No Calified Vars	OPT.COBOL.AvoidNoCalifiedVars	AvoidNoCalifiedVars: Variable with repeated ID detected
Info	Avoid No Rewind In Sequential Files	OPT.COBOL.AvoidNoRewindInSequentialFiles	AvoidNoRewindInSequentialFiles: Do not use NO REWIND clause in sequential files
Info	Avoid Recover Equal Fields	OPT.COBOL.AvoidRecoverEqualFields	AvoidRecoverEqualFields: Avoid recovering fields that are conditioned by equal
Info	Avoid Repeat Calls	OPT.COBOL.AvoidRepeatCalls	AvoidRepeatCalls: Avoid calling to the same routine several times
Info	Avoid Search Small Working	OPT.COBOL.AvoidSearchSmallWorking	AvoidSearchSmallWorking: For seeking in tables WORKING with less than 50 elements use the SEARCH statement
Info	Avoid Sentence According Size Table Small	OPT.COBOL.AvoidSentenceAccordingSizeTableSmall	AvoidSentenceAccordingSizeTableSmall: Detected a statement on a small table (No. pages < {})
Info	Change Cursor To Select	OPT.COBOL.ChangeCursorToSelect	ChangeCursorToSelect: Defined cursor can be transformed into a SELECT
Info	Check88 Vars	OPT.COBOL.Check88Vars	Check88Vars: It is recommended using level 88 for variables used in conditional statements
Info	Check Cols Not Modify	OPT.COBOL.CheckColsNotModify	CheckColsNotModify: Do not put columns whose value has not been changed in the UPDATE statement

Severity	Contrast rule	Engine rule ID	Description
Info	Check Complete Insert	OPT.COBOL.CheckCompleteInsert	CheckCompleteInsert: The INSERT statement should be coded with all the variables and in the same order as defined in DECLARE
Info	Check Cursor For Update	OPT.COBOL.CheckCursorForUpdate	CheckCursorForUpdate: The columns retrieved in the FOR UPDATE clause must match those specified in the SET clause of the WHERE CURRENT OF
Info	Check Delete For Update	OPT.COBOL.CheckDeleteForUpdate	CheckDeleteForUpdate: When using cursors, to delete rows with DELETE WHERE CURRENT OF, the FOR UPDATE clause must have a single column
Info	Check Fetch And Cursor	OPT.COBOL.CheckFetchAndCursor	CheckFetchAndCursor: The FETCH statement must include the same columns in the same order they appear in the statement of the cursor
Info	Check File Operations	OPT.COBOL.CheckFileOperations	CheckFileOperations: Do not use OPEN, READ, and WRITE operations more than once in a program
Info	Check Filestatus After File Access	OPT.COBOL.CheckFilestatusAfterFileAccess	CheckFilestatusAfterFileAccess: Check the FILE STATUS variable after each file access
Info	Check List Prefetch	OPT.COBOL.CheckListPrefetch	CheckListPrefetch: LIST PREFETCH has been detected in DB2 access
Info	Check Type Operations	OPT.COBOL.CheckTypeOperations	CheckTypeOperations: Fields involved in arithmetic operations must be defined as COMP or COMP-3 and have the same length
Info	Check Updt For Updt	OPT.COBOL.CheckUpdtForUpdt	CheckUpdtForUpdt: The updated columns with UPDATE must be the same as declared in the FOR UPDATE clause
Info	Check Working Structure	OPT.COBOL.CheckWorkingStructure	CheckWorkingStructure: WORKING tables must be defined at the end of the WORKING and before the declaration of cursors
Info	C S I M	OPT.COBOL.COBBP.CSIM	CSIM: Do not use comparison symbols, use the corresponding name instead
Info	S O C	OPT.COBOL.COBBP.SOC	SOC: A single OPEN / CLOSE per file
Info	N O Don't use masterpage files	OPT.COBOL.COD_COBOL.NOMP	NOMP: Paragraph names must begin with a user-specified prefix
Info	N O M S	OPT.COBOL.COD_COBOL.NOMS	NOMS: Procedural section names must follow naming convention
Info	N P A R	OPT.COBOL.COD_COBOL.NPAR	NPAR: Paragraph names must follow naming convention
Info	N V W S	OPT.COBOL.COD_COBOL.NVWS	NVWS: The names of the variables and constants of the WORKING-STORAGE must follow the user-specified prefixes
Info	Cols Should Be Used	OPT.COBOL.ColsShouldBeUsed	ColsShouldBeUsed: All columns declared in the SELECT of a cursor defined WITH ROWSET POSITIONING must be used later in the program
Info	Control Num Rows	OPT.COBOL.ControlNumRows	ControlNumRows: When use the MULTIROW option, rowset size should never be more than 200 rows

Severity	Contrast rule	Engine rule ID	Description
Info	Count Valid Lines	OPT.COBOL.CountValidLines	CountValidLines: Small programs, with less than a certain number of lines, are recommended
Info	Cursors At The End Of Working	OPT.COBOL.CursorsAtTheEndOfWorking	CursorsAtTheEndOfWorking: NULL indicator, for SQL, must be PIC S9 (4) COMP
Info	Display At End	OPT.COBOL.DisplayAtEnd	DisplayAtEnd: Using DISPLAY is only allowed by program end or ABEND
Info	Do Not Include SQLca Without Db2	OPT.COBOL.DoNotIncludeSqlcaWithoutDb2	DoNotIncludeSqlcaWithoutDb2: Exclude SQLCA if no DB2 accesses in the program
Info	Do Not Use Comp2	OPT.COBOL.DoNotUseComp2	DoNotUseComp2: Using COMP-2 is not allowed
Info	Do Not Use Dclgen At Level01	OPT.COBOL.DoNotUseDclgenAtLevel01	DoNotUseDclgenAtLevel01: It is not allowed the use of DCLGEN to Level 01
Info	Do Not Use Filler At Level01	OPT.COBOL.DoNotUseFillerAtLevel01	DoNotUseFillerAtLevel01: Use of FILLER at level 01 is not recommended
Info	Do Not Use Linage Clause	OPT.COBOL.DoNotUseLinageClause	DoNotUseLinageClause: Use of clause LINAGE is not allowed
Info	Do Not Use Many Files	OPT.COBOL.DoNotUseManyFiles	DoNotUseManyFiles: The use of more than 10 files in a program is not allowed
Info	Do Not Use Select To Check A Row	OPT.COBOL.DoNotUseSelectToCheckARow	DoNotUseSelectToCheckARow: Do not use SELECT / FETCH to verify the existence of a row for later reading or updating
Info	Do Not Use Static Calls Routines	OPT.COBOL.DoNotUseStaticCallsRoutines	DoNotUseStaticCallsRoutines: Do not make calls to routines statically
Info	Duplicated Data Access	OPT.COBOL.DuplicatedDataAccess	DuplicatedDataAccess: Duplicated SQL accesses were detected
Info	B T A	OPT.COBOL.FIA_COBOL.BTA	BTA: Do not use TEST AFTER inside a loop
Info	C E R M	OPT.COBOL.FIA_COBOL.CERM	CERM: Adjust type and length of source and target in a MOVE statement
Info	D E C P	OPT.COBOL.FIA_COBOL.DCEP	DECP: DECIMAL POINT IS COMMA must be specified
Info	I I N I	OPT.COBOL.FIA_COBOL.IINI	IINI: Variable initialization with INITIALIZE
Info	M E R G	OPT.COBOL.FIA_COBOL.MERG	MERG: Avoid MERGE statement
Info	N O S R	OPT.COBOL.FIA_COBOL.NOSR	NOSR: Use GOBACK instead of STOP RUN
Info	File Without Filestatus	OPT.COBOL.FileWithoutFilestatus	FileWithoutFilestatus: File definition without FILESTATUS or FILESTATUS bad defined
Info	Group Open And Close	OPT.COBOL.GroupOpenAndClose	GroupOpenAndClose: Opening and closing files should be grouped in one OPEN and CLOSE
Info	Incorrect Indicator Defined	OPT.COBOL.IncorrectIndicatorDefined	IncorrectIndicatorDefined: NULL indicator, for SQL, must be PIC S9 (4) COMP
Info	Initialize Var Level01	OPT.COBOL.InitializeVarLevel01	InitializeVarLevel01: Use of INITIALIZE is recommended for variables level 01
Info	C I N W	OPT.COBOL.MAN_COBOL.CINW	CINW: Use standard naming for WORKING-STORAGE variables

Severity	Contrast rule	Engine rule ID	Description
Info	C M F D	OPT.COBOLEMAN_COBOLECMFD	CMFD: Before an FD (file descriptor) it is mandatory to add a comment explaining the file purpose and encoded information
Info	E D P	OPT.COBOLEMAN_COBOLEDPE	EDP: Avoid program descriptions larger than N lines
Info	I D O P	OPT.COBOLEMAN_COBOLEIDOP	IDOP: Indent properly instruction operands
Info	I I T	OPT.COBOLEMAN_COBOLEIIT	IIT: Add a prefix for Cobol tables and a suffix for their indexes
Info	I R T	OPT.COBOLEMAN_COBOLEIRT	IRT: Naming convention for structured tables and their records
Info	L F D	OPT.COBOLEMAN_COBOLELFD	LFD: Leave a blank line between file descriptors (FD)
Info	L I S E	OPT.COBOLEMAN_COBOLELISE	LISE: Avoid too long procedural sections
Info	L P R E	OPT.COBOLEMAN_COBOLELPRE	LPRE: Paragraph name length should be between certain limits
Info	L T E R	OPT.COBOLEMAN_COBOLELTER	LTER: Use capital letters for code
Info	M N M X	OPT.COBOLEMAN_COBOLEMNMX	MNMX: Use uppercase for code and lowercase for comments
Info	M S E C	OPT.COBOLEMAN_COBOLEMSEC	MSEC: Avoid programs with too many procedural sections
Info	N C M A	OPT.COBOLEMAN_COBOLENCMA	NCMA: Avoid commas for separating parameters in DISPLAY statements
Info	N C S W	OPT.COBOLEMAN_COBOLENCWS	NCSW: Avoid nesting EVALUATE too deeply
Info	N R E G	OPT.COBOLEMAN_COBOLENREG	NREG: FD records naming convention
Info	O V W S	OPT.COBOLEMAN_COBOLEOVWS	OVWS: WORKING-STORAGE variables must follow a particular order
Info	P D E S	OPT.COBOLEMAN_COBOLEPDES	PDES: Document program in code comments before PROGRAM-ID
Info	P I F	OPT.COBOLEMAN_COBOLEPIF	PIF: Limit the number of sentences in IF or ELSE blocks
Info	P I N I	OPT.COBOLEMAN_COBOLEPINI	PINI: The first paragraph in PROCEDURE DIVISION must follow a standard name
Info	P L I N	OPT.COBOLEMAN_COBOLEPLIN	PLIN: PIC clauses should be properly aligned
Info	P P A R	OPT.COBOLEMAN_COBOLEPPAR	PPAR: Paragraphs must contain an empty end line with a single dot
Info	P R I D	OPT.COBOLEMAN_COBOLEPRID	PRID: PROGRAM-ID must be the same as the program name (filename, without extension)
Info	P W E	OPT.COBOLEMAN_COBOLEPWE	PWE: Use PERFORM instead of EVALUATE ... WHEN with many nested lines
Info	V L I N	OPT.COBOLEMAN_COBOLEVLIN	VLIN: Literals of the VALUE clauses for each level in data declarations should be aligned
Info	No Optional In File Control	OPT.COBOLE.NoOptionalInFileControl	NoOptionalInFileControl: The OPTIONAL clause is not allowed in FILE-CONTROL
Info	Obligatory End Read	OPT.COBOLE.ObligatoryEndRead	ObligatoryEndRead: Each READ must be ended with its corresponding END-READ
Info	Obligatory End Search	OPT.COBOLE.ObligatoryEndSearch	ObligatoryEndSearch: Each SEARCH must be ended with its corresponding END-SEARCH

Severity	Contrast rule	Engine rule ID	Description
Info	B I U S	OPT.COBOLO.YR_COBOLO.BIUS	BIUS: Do not use BINARY, COMP or COMP-4 with variables/constants with more than 10 digits
Info	C D I N	OPT.COBOLO.YR_COBOLO.CDIN	CDIN: Always CALL subprograms passing parameters BY REFERENCE
Info	C O P Y	OPT.COBOLO.YR_COBOLO.COPY	COPY: Standard copybooks must be included
Info	D U P	OPT.COBOLO.YR_COBOLO.DUP	DUP: Avoid DISPLAY...UPON CONSOLE
Info	I N D B	OPT.COBOLO.YR_COBOLO.INDB	INDB: Variables used as table indexes must be of type S9(2) COMP or S9(4) COMP
Info	M C O R	OPT.COBOLO.YR_COBOLO.MCOR	MCOR: Avoid CORRESPONDING clause in MOVE, ADD and SUBTRACT statements
Info	N C P Y	OPT.COBOLO.YR_COBOLO.NCPY	NCPY: Avoid including copybooks not present in a standard set
Info	N D I S	OPT.COBOLO.YR_COBOLO.NDIS	NDIS: Avoid arithmetic operations on DISPLAY variables
Info	N N S	OPT.COBOLO.YR_COBOLO.NNS	NNS: Avoid usage of NEXT SENTENCE
Info	N O R E	OPT.COBOLO.YR_COBOLO.NORE	NORE: Do not use RELEASE
Info	Ocurrences Table Elements	OPT.COBOLO.YR_COBOLO.OcurrencesTableElements	OcurrencesTableElements: Optimize access to table elements
Info	P A R N	OPT.COBOLO.YR_COBOLO.PARN	PARN: Avoid routines with too many parameters
Info	P D I M	OPT.COBOLO.YR_COBOLO.PDIM	PDIM: With PACKED-DECIMAL / COMP-3, use less than 16 digits, and even (for signed) or odd (for unsigned) digits
Info	S I B Y	OPT.COBOLO.YR_COBOLO.SIBY	SIBY: Use SYNCHRONIZED with binary (BINARY,COMP,COMP-4 and COMP-5) fields
Info	S O R T	OPT.COBOLO.YR_COBOLO.SORT	SORT: Avoid SORT statement
Info	T I M E	OPT.COBOLO.YR_COBOLO.TIME	TIME: Access only once to system variables DATE, DAY, DAY-OF-WEEK, TIME, CENTURY-DATE, CENTURY-DAY, CURRENT-DATE
Info	Perform Times With Memory Tables	OPT.COBOLO.PerformTimesWithMemoryTables	PerformTimesWithMemoryTables: Using PERFORM N TIMES is only allowed with in-memory tables
Info	Perform Thru With Exit	OPT.COBOLO.PerformThruWithExit	PerformThruWithExit: Each PERFORM THRU must have its corresponding paragraph with EXIT
Info	A P I C	OPT.COBOLO.RG_COBOLO.APIC	APIC: Use parenthesis in PIC clauses instead of repeated XX, AA or 99
Info	C L A U	OPT.COBOLO.RG_COBOLO.CLAU	CLAU: Do not include deprecated paragraphs in IDENTIFICATION DIVISION (like DATE-COMPILED, DATE-WRITTEN, INSTALLATION, AUTHOR, SECURITY)
Info	C P I C	OPT.COBOLO.RG_COBOLO.CPIC	CPIC: Use PIC instead of PICTURE
Info	F N F	OPT.COBOLO.RG_COBOLO.FNF	FNF: Use a specific level and data name for the first entry in WORKING-STORAGE SECTION
Info	I N B Y	OPT.COBOLO.RG_COBOLO.INBY	INBY: Use INDEXED BY clause with Cobol tables (fields with OCCURS)

Severity	Contrast rule	Engine rule ID	Description
Info	M V D	OPT.COBOL.RG_COBOL.MVD	MVD: Do not use MOVE with constant literals, use a named constant field instead
Info	N77	OPT.COBOL.RG_COBOL.N77	N77: Do not use 77 levels
Info	N I	OPT.COBOL.RG_COBOL.NI	NI: Use odd levels for data entries in WORKING-STORAGE SECTION
Info	N L P	OPT.COBOL.RG_COBOL.NLP	NLP: Do not use literals in PROCEDURE DIVISION sentences
Info	N N IV	OPT.COBOL.RG_COBOL.NNIV	NNIV: In DATA DIVISION, all data entry levels should be 01 or multiple of 5
Info	N P N T	OPT.COBOL.RG_COBOL.NPNT	NPNT: Do not write dots in sentences that do not require them
Info	N T H N	OPT.COBOL.RG_COBOL.NTHN	NTHN: Do not use THEN in an IF sentence
Info	Section End Doesnt Exist	OPT.COBOL.SectionEndDoesntExist	SectionEndDoesntExist: Section end does not exist
Info	SQL Statements Not Executed	OPT.COBOL.SqlStatementsNotExecuted	SqlStatementsNotExecuted: Existence of SQL statements that are not executed
Info	Too Much Call	OPT.COBOL.TooMuchCall	TooMuchCall: Exceeded the maximum allowed number of calls to routines
Info	Use Index Field To Check A Row	OPT.COBOL.UseIndexFieldToCheckARow	UseIndexFieldToCheckARow: If you need to know whether there is a row, search it selecting a field that is index
Info	Use Varying Only With Tables	OPT.COBOL.UseVaryingOnlyWithTables	UseVaryingOnlyWithTables: Using PERFORM VARYING is only allowed with inmemory tables
Info	Avoid Non Qualified Joins	OPT.COBOL.SQL_COBOL.AvoidNonQualifiedJoins	AvoidNonQualifiedJoins: Make the type of join explicit
Info	Cobol Password In Comment	OPT.COBOL.SEC.Cobol_PasswordInComment	Cobol_PasswordInComment: Avoid placing passwords and other sensitive info in code comments
Info	Cobol Privacy Violation	OPT.COBOL.SEC.Cobol_PrivacyViolation	Cobol_PrivacyViolation: Exposure of Private Information ('Privacy Violation')
Low	Access In Loop More Than One Index	OPT.COBOL.AccessInLoopMoreThanOneIndex	AccessInLoopMoreThanOneIndex: Access detected in a loop which use more than one index to resolve the access to a table (no. pages >
Low	Access In Loop Without Index	OPT.COBOL.AccessInLoopWithoutIndex	AccessInLoopWithoutIndex: Access detected in a loop in which there are not informed index fields in a table WHERE (no. pages >
Low	Avoid Select With Low Conditions	OPT.COBOL.AvoidSelectWithLowConditions	AvoidSelectWithLowConditions: Avoid using 'SELECT function' with low discrimininity conditions in WHERE clause
Low	Avoid Access Not Indexed Table Medium	OPT.COBOL.AvoidAccessNotIndexedTableMedium	AvoidAccessNotIndexedTableMedium: Detected an access to a table size medium (7 < no. pages < 10.000) in which the first index fields or DB2 can not use them are not reported
Low	Avoid Access Without Index Medium	OPT.COBOL.AvoidAccessWithoutIndexMedium	AvoidAccessWithoutIndexMedium: Access is detected in which there is no index fields in the WHERE informed on a medium size table (7 < no. pages < 10.000)

Severity	Contrast rule	Engine rule ID	Description
Low	Avoid Big Tables	OPT.COBOL.AvoidBigTables	AvoidBigTables: Definition of a table in LINKAGE or WORKING with a very large size or too many elements
Low	Avoid Bulk Updates In A Sentence	OPT.COBOL.AvoidBulkUpdatesInASentence	AvoidBulkUpdatesInASentence: It is not allowed bulk updates using a SQL statement
Low	Avoid Cancel	OPT.COBOL.AvoidCancel	AvoidCancel: Using clause CANCEL is not allowed
Low	Avoid On Size Error	OPT.COBOL.AvoidOnSizeError	AvoidOnSizeError: Do not use ON SIZE ERROR
Low	Avoid Select Ast Check Rows	OPT.COBOL.AvoidSelectAstCheckRows	AvoidSelectAstCheckRows: Do not use SELECT COUNT(*) to verify the existence of rows
Low	Avoid Sentence According Size Table Medium	OPT.COBOL.AvoidSentenceAccordingSizeTableMedium	AvoidSentenceAccordingSizeTableMedium: Detected a sentence with a medium sized table (7 < no. pages < 10.000) that uses more than one index to resolve access
Low	Avoid Sentence According Size Table Big	OPT.COBOL.AvoidSentenceAccordingSizeTableBig	AvoidSentenceAccordingSizeTableBig: Detected a sentence with a big table (No. pages > {})
Low	Check Cursor Instead Of Statements	OPT.COBOL.CheckCursorInsteadOfStatements	CheckCursorInsteadOfStatements: Use CURSOR FOR UPDATE instead SELECT and UPDATE/DELETE
Low	Check Cursor Positioning Fetch	OPT.COBOL.CheckCursorPositionningFetch	CheckCursorPositionningFetch: If a cursor is defined WITH ROWSET POSITIONING, the FETCH for that cursor must be defined with the NEXT ROWSET clause and viceversa
Low	Check Deq After Enq	OPT.COBOL.CheckDeqAfterEnq	CheckDeqAfterEnq: When the ENQ command is used it must issue the DEQ command as soon as possible
Low	Check Func Columns	OPT.COBOL.CheckFuncColumns	CheckFuncColumns: It is not allowed to use functions on columns in the WHERE clause of SQL statements
Low	Check Func Host Vars In Where	OPT.COBOL.CheckFuncHostVarsInWhere	CheckFuncHostVarsInWhere: It is not allowed to use functions on HOST variables in the WHERE clause of SQL statements
Low	Check Low Volume Tables Very Accessed	OPT.COBOL.CheckLowVolumeTablesVeryAccessed	CheckLowVolumeTablesVeryAccessed : DB2 tables of low volume and very accessed must be copied in WORKING at the beginning of the program execution
Low	Check Order Sentences	OPT.COBOL.CheckOrderSentences	CheckOrderSentences: The statement triggers a process management DB2 expensive due to the high number of selected rows
Low	Check Return In Cics	OPT.COBOL.CheckReturnInCics	CheckReturnInCics: Always check the return code of CICS statements to avoid ABENDs in transactions
Low	Check Search At End	OPT.COBOL.CheckSearchAtEnd	CheckSearchAtEnd: In the SEARCH statement, use the AT END clause
Low	Check Vars To Read	OPT.COBOL.CheckVarsToRead	CheckVarsToRead: Do not use in READ the file record, or a WORKING variable smaller than the file record.
Low	Check Vars To Write	OPT.COBOL.CheckVarsToWrite	CheckVarsToWrite: Do not use in WRITE the file record, or a WORKING variable greater than the record size.

Severity	Contrast rule	Engine rule ID	Description
Low	Check Where Like	OPT.COBOL.CheckWhereLike	CheckWhereLike: Avoid using LIKE '%' and LIKE ' _ '
Low	Check Write Stmt	OPT.COBOL.CheckWriteStmt	CheckWriteStmt: Do not use AFTER or BEFORE in WRITE operation
Low	IP L	OPT.COBOL.COBBP.IPL	IPL: One statement per line
Low	P V A C	OPT.COBOL.COBBP.PVAC	PVAC: Avoid empty paragraphs
Low	C I N	OPT.COBOL.COD_COBOL.CIN	CIN: Called subprogram name should follow naming convention
Low	Data Division	OPT.COBOL.COD_COBOL.DataDivision	DataDivision: There are data definitions outside DATA DIVISION
Low	Working Storage Var Names	OPT.COBOL.COD_COBOL.WorkingStorageVarNames	WorkingStorageVarNames: WORKING STORAGE variables and constants name format
Low	Type Time	OPT.COBOL.COD_COBOL.TypeTime	TypeTime: TIMESTAMP and TIME variables format
Low	Do Not Open In Bucle	OPT.COBOL.DoNotOpenInBucle	DoNotOpenInBucle: Avoid multiple Open/Close for the same file in the program
Low	Do Not Repeat Access	OPT.COBOL.DoNotRepeatAccess	DoNotRepeatAccess: Recover data from a table using a single access to it
Low	Do Not Use Return Code	OPT.COBOL.DoNotUseReturnCode	DoNotUseReturnCode: Not allowed to use the RETURN-CODE variable
Low	Do Not Use Rewrite In Sequential	OPT.COBOL.DoNotUseRewriteInSequential	DoNotUseRewriteInSequential: You are not allowed to use REWRITE on sequential files
Low	C B U C	OPT.COBOL.FIA_COBOL.CBUC	CBUC: Do not use equal as end-loop condition
Low	C F D	OPT.COBOL.MAN_COBOL.CFD	CFD: Use COPYs for file/sort record definitions
Low	F L C B	OPT.COBOL.MAN_COBOL.FLCB	FLCB: Separate paragraphs with empty comments
Low	I A I D	OPT.COBOL.MAN_COBOL.IAID	IAID: Include AUTHOR field in Division Identification
Low	I I E	OPT.COBOL.MAN_COBOL.IIE	IIE: Avoid incorrect ELSE indentation
Low	I I I	OPT.COBOL.MAN_COBOL.III	III: Use correct indentation within IF statements
Low	I I R	OPT.COBOL.MAN_COBOL.IIR	IIR: Indent READ statements properly
Low	I I R W	OPT.COBOL.MAN_COBOL.IIRW	IIRW: Indent REWRITE statements properly
Low	I I W	OPT.COBOL.MAN_COBOL.IIW	IIW: Indent WRITE sentences properly
Low	N C F D	OPT.COBOL.MAN_COBOL.NCFD	NCFD: Do not use COPYs for file/sort descriptors
Low	Register Validation With Select	OPT.COBOL.RegisterValidationWithSelect	RegisterValidationWithSelect: Validating if a record exists by SELECT
Low	W D	OPT.COBOL.RG_COBOL.WD	WD: Working-storage section definitions order
Low	Avoid Insert Without Fields Specification	OPT.COBOL.SQL_COBOL.AvoidInsertWithoutFieldsSpecification	AvoidInsertWithoutFieldsSpecification: Every INSERT statement must include the field specification (i.e : INSERT INTO table(column1,column2) VALUES (value1,value2))
Low	Avoid Qualified Tables In Queries	OPT.COBOL.SQL_COBOL.AvoidQualifiedTablesInQueries	AvoidQualifiedTablesInQueries: Table names should not be qualified in queries
Low	Qualified Tables In Queries	OPT.COBOL.SQL_COBOL.QualifiedTablesInQueries	QualifiedTablesInQueries: Every table referenced in the query must be qualified

Severity	Contrast rule	Engine rule ID	Description
Low	Use Search All	OPT.COBOL.UseSearchAll	UseSearchAll: For seeking in tables WORKING with more than 50 elements use the SEARCH ALL statement
Medium	Close Statements With Nested Body	OPT.COBOL.COBBP.CloseStatementsWithNestedBody	CloseStatementsWithNestedBody: Close statements that may include code blocks with explicit END delimiter
Medium	To End Paragraph	OPT.COBOL.COBBP.ToEndParagraph	ToEndParagraph: Check that top-level paragraphs have an exit paragraph
Medium	Avoid Arithmetic Operations In If	OPT.COBOL.FIA_COBOL.AvoidArithmeticOperationsInIf	AvoidArithmeticOperationsInIf: Check that there are no arithmetic operations in the condition for IF statements
Medium	Obligatory End Evaluate	OPT.COBOL.FIA_COBOL.ObligatoryEndEvaluate	ObligatoryEndEvaluate: Check that every EVALUATE is closed by an END-EVALUATE
Medium	Avoid Explicit Data In Linkage	OPT.COBOL.MAN_COBOL.AvoidExplicitDataInLinkage	AvoidExplicitDataInLinkage: LINKAGE SECTION should not contain explicit data description entries
Medium	Avoid Procedural Copybook	OPT.COBOL.MAN_COBOL.AvoidProceduralCopybook	AvoidProceduralCopybook: Avoid including copybooks to share procedural code
Medium	Avoid Too Deep Perform Chains	OPT.COBOL.MAN_COBOL.AvoidTooDeepPerformChains	AvoidTooDeepPerformChains: Avoid too deep PERFORM chains
Medium	C C A L	OPT.COBOL.MAN_COBOL.CCAL	CCAL: Document all program calls immediately before the call
Medium	Copy Book With Data Or Procedures	OPT.COBOL.MAN_COBOL.CopyBookWithDataOrProcedures	CopyBookWithDataOrProcedures: Copybooks should contain data definitions or procedural code only
Medium	H I C E	OPT.COBOL.MAN_COBOL.HICE	HICE: Avoid programs with too many GOTO
Medium	I N 0 1	OPT.COBOL.MAN_COBOL.IN01	IN01: Comment any top-level variable (level 01)
Medium	N A M I N G P R O G R A M I D	OPT.COBOL.MAN_COBOL.NAMINGPROGRAMID	NAMINGPROGRAMID: Program name must follow a naming standard
Medium	P I C	OPT.COBOL.MAN_COBOL.PIC	PIC: Program name variables in CALL statements must follow naming convention
Medium	Reference Modifier	OPT.COBOL.OYR_COBOL.ReferenceModifier	ReferenceModifier: Position and length variables in VAR(position:length) should have short binary types
Medium	V O D T	OPT.COBOL.OYR_COBOL.VODT	VODT: Do not perform arithmetic operations on variables of different types
Medium	C W S V	OPT.COBOL.RG_COBOL.CWSV	CWSV: Avoid entries in WORKING-STORAGE SECTION without initial value
Medium	D P I C	OPT.COBOL.RG_COBOL.DPIC	DPIC: Include DECIMAL-POINT IS COMMA when there is at least one edited field or constant decimal in the programme
Medium	I F W	OPT.COBOL.RG_COBOL.IFW	IFW: Convention for working end and beginning
Medium	N E P	OPT.COBOL.RG_COBOL.NEP	NEP: Do not use EXIT
Medium	Cobol System Information Leak	OPT.COBOL.SEC.Cobol_SystemInformationLeak	Cobol_SystemInformationLeak: Avoid dumping system info (typically for debugging) in production code

Severity	Contrast rule	Engine rule ID	Description
Medium	Poor Error Handling	OPT.COBOL.SEC.PoorErrorHandling	PoorErrorHandling: Ignoring error conditions may allow an attacker to induce unexpected behavior unnoticed
Medium	Avoid Natural Joins	OPT.COBOL.SQL_COBOL.AvoidNaturalJoins	AvoidNaturalJoins: NATURAL JOINS are buggy and unmaintenable
Medium	Avoid Select Asterisk	OPT.COBOL.SQL_COBOL.AvoidSelectAsterisk	AvoidSelectAsterisk: Do not use SELECT *
Medium	Prefer On Over Using	OPT.COBOL.SQL_COBOL.PreferOnOverUsing	PreferOnOverUsing: Replace Using clause for its equivalent On counterpart
Medium	Detect Implicit Joins	OPT.COBOL.SQL_COBOL.DetectImplicitJoins	DetectImplicitJoins: Never use implicit JOINS
Medium	Avoid Too Many Joins	OPT.COBOL.SQL_COBOL.AvoidTooManyJoins	AvoidTooManyJoins: Avoid queries with too many JOINS
Medium	Avoid Queries On Many Tables	OPT.COBOL.SQL_COBOL.AvoidQueriesOnManyTables	AvoidQueriesOnManyTables: Avoid JOIN queries referencing too many tables
Medium	Avoid Nested Selects	OPT.COBOL.SQL_COBOL.AvoidNestedSelects	AvoidNestedSelects: Avoid nested selects
Medium	Cobol Password With Weak Crypto	OPT.COBOL.SEC.Cobol_PasswordWithWeakCrypto	Cobol_PasswordWithWeakCrypto: Weak Cryptography for Passwords

CPP Scan rules

Contrast Scan supports these rules for CPP.

Severity	Contrast rule	Engine rule ID	Description
Critical	Avoid Comp Diff Types	OPT.CPP.AvoidCompDiffTypes	AvoidCompDiffTypes: Do not compare variables of different basic types
Critical	Adding or subtracting an integer to a pointer if resulting value does not refer to a valid array element	OPT.CPP.CERTC.ARR38	ARR38: Do not add or subtract an integer to a pointer if the resulting value does not refer to a valid array element
Critical	NULL Pointer Dereference	OPT.CPP.CERTC.EXP34	EXP34: NULL Pointer Dereference
Critical	Do not access freed memory	OPT.CPP.CERTC.MEM30	MEM30: Do not access freed memory (Use of freed memory)
Critical	Freeing Memory not on the Heap	OPT.CPP.CERTC.MEM34	MEM34: Free of Memory not on the Heap
Critical	Do not replace secure functions with less secure functions	OPT.CPP.CERTC.PRE09	PRE09: Do not replace secure functions with less secure functions
Critical	Signal Handler Use of a Non-reentrant Function	OPT.CPP.CERTC.SIG30	SIG30: Signal Handler Use of a Non-reentrant Function
Critical	Signal Handler Use of a Non-reentrant Function	OPT.CPP.CERTC.SIG32	SIG32: Signal Handler Use of a Non-reentrant Function

Severity	Contrast rule	Engine rule ID	Description
Critical	Guarantee that storage for strings has sufficient space	OPT.CPP.CERTC.STR31	STR31: Guarantee that storage for strings has sufficient space for character data and the null terminator
Critical	Size wide character strings correctly	OPT.CPP.CERTC.STR33	STR33: Size wide character strings correctly
Critical	Do not copy data from an unbounded source to a fixed-length array	OPT.CPP.CERTC.STR35	STR35: Do not copy data from an unbounded source to a fixed-length array
Critical	Destructors Must Be Noexcept	OPT.CPP.COREGL.DestructorsMustBeNoexcept	DestructorsMustBeNoexcept: Destructor must be noexcept.
Critical	Multiple Mutexes Acquired On Separate Locks	OPT.CPP.COREGL.MultipleMutexesAcquiredOnSeparateLocks	MultipleMutexesAcquiredOnSeparateLocks: Multiple mutexes should be acquired in a single lock
Critical	Temporary RAIIObj	OPT.CPP.COREGL.TemporaryRAIIObj	TemporaryRAIIObj: Temporary RAII objects should be managed with RAII
Critical	Wait Without Condition	OPT.CPP.COREGL.WaitWithoutCondition	WaitWithoutCondition: Call to "std::condition_variable::wait()" without a condition
Critical	Check Return In Public Functions	OPT.CPP.CheckReturnInPublicFunctions	CheckReturnInPublicFunctions: Functions should check return values and return a pointer or reference to local variables
Critical	Number Args In Calls Must Match Formal Params	OPT.CPP.MISRAC.NumberArgsInCallsMustMatchFormalParams	NumberArgsInCallsMustMatchFormalParams: MISRAC 16.6: The number of arguments passed to a function shall match the number of parameters
Critical	Avoid Throw Exception In Destructor	OPT.CPP.AvoidThrowExceptionInDestructor	AvoidThrowExceptionInDestructor: Never throw an exception from a destructor
Critical	Braces In Array Delete	OPT.CPP.BracesInArrayDelete	BracesInArrayDelete: Arrays allocated with "new[]" must be deallocated with "delete[]"
Critical	Class With New Must Define Copy Cons And Assignment Op	OPT.CPP.ClassWithNewMustDefineCopyConsAndAssignmentOp	ClassWithNewMustDefineCopyConsAndAssignmentOp: Classes that allocate memory in data members must define copy constructor and assignment operator
Critical	No Base Class Without Virtual Destructor	OPT.CPP.NoBaseClassWithoutVirtualDestructor	NoBaseClassWithoutVirtualDestructor: Derivatives must have a virtual destructor in every base class.
Critical	No Global Objects In Const And Destr	OPT.CPP.NoGlobalObjectsInConstAndDestr	NoGlobalObjectsInConstAndDestr: Avoid using global objects in constructors and destructors
Critical	No Member In Class Definition	OPT.CPP.NoMemberInClassDefinition	NoMemberInClassDefinition: No member functions should be defined within the class definition
Critical	No Virtual Method Calls In Const Or Destr	OPT.CPP.NoVirtualMethodCallsInConstOrDestr	NoVirtualMethodCallsInConstOrDestr: Avoid calling virtual functions from constructors or destructors
Critical	Virtual Destructor If Virtual Method	OPT.CPP.VirtualDestructorIfVirtualMethod	VirtualDestructorIfVirtualMethod: Avoid classes with at least one virtual method and without a virtual destructor

Severity	Contrast rule	Engine rule ID	Description
Critical	Write Operator Delete With Operator New	OPT.CPP.WriteOperatorDeleteWithOperatorNew	WriteOperatorDeleteWithOperatorNew: Im 'delete' if there is 'new' implemented
Critical	Anonymous Ldap Bind	OPT.CPP.SEC.AnonymousLdapBind	AnonymousLdapBind: Access Control - An LDAP Bind
Critical	Path Traversal	OPT.CPP.SEC.PathTraversal	PathTraversal: Avoid non-neutralized user- input composed in a pathname to a resourc
Critical	Static Database Connection	OPT.CPP.SEC.StaticDatabaseConnection	StaticDatabaseConnection: Static database connection / session
Critical	Unsafe Chroot	OPT.CPP.SEC.UnsafeChroot	UnsafeChroot: Unsafe chroot call.
Critical	Exclude unsanitized input	OPT.CPP.CERTC.FIO30	FIO30: Exclude unsanitized user input from strings
Critical	Sanitize data passed to sensitive subsystems	OPT.CPP.CERTC.STR02	STR02: Sanitize data passed to sensitive s
Critical	Connection String Parameter Pollution	OPT.CPP.SEC.ConnectionStringParameterPollution	ConnectionStringParameterPollution: Conn polluted with untrusted input
Critical	DoS Regexp	OPT.CPP.SEC.DoSRegexp	DoSRegexp: Prevent denial of service attac malicious regular expression ('Regexp Injec
Critical	Ldap Injection	OPT.CPP.SEC.LdapInjection	LdapInjection: Avoid non-neutralized user- input in LDAP search filters
Critical	No SQL Injection	OPT.CPP.SEC.NoSQLInjection	NoSQLInjection: Improper neutralization of elements in data query logic (NoSQL inject
Critical	Process Control	OPT.CPP.SEC.ProcessControl	ProcessControl: Do not load executables o from untrusted sources
Critical	SQL Injection	OPT.CPP.SEC.SqlInjection	SqlInjection: Improper Neutralization of Sp Elements used in an SQL Command ('SQL
Critical	Xml Entity Injection	OPT.CPP.SEC.XmlEntityInjection	XmlEntityInjection: XML entity injection
Critical	Hardcoded Crypto Key	OPT.CPP.SEC.HardcodedCryptoKey	HardcodedCryptoKey: Hardcoded cryptogr
High	Avoid Auto Ptr	OPT.CPP.AvoidAutoPtr	AvoidAutoPtr: Avoid auto_ptr.
High	Avoid Calling Too Many Other Components	OPT.CPP.AvoidCallingTooManyOtherComponents	AvoidCallingTooManyOtherComponents: A components calling too many other compo
High	Avoid Excessive Nested Statements	OPT.CPP.AvoidExcessiveNestedStatements	AvoidExcessiveNestedStatements: Avoid a control flow statements nesting depth
High	Avoid Object Instantiation Into Loops	OPT.CPP.AvoidObjectInstantiationIntoLoops	AvoidObjectInstantiationIntoLoops: Avoid o instantiation into loops
High	Avoid Signal Management Functions	OPT.CPP.AvoidSignalManagmentFunctions	AvoidSignalManagmentFunctions: Avoid u management functions
High	Avoid Structures	OPT.CPP.AvoidStructures	AvoidStructures: Avoid using certain kinds objects (struct, union, VARIANT)
High	Avoid Too Complex Functions	OPT.CPP.AvoidTooComplexFunctions	AvoidTooComplexFunctions: Avoid using f high cyclomatic complexity values
High	Avoid Too Complex Programs	OPT.CPP.AvoidTooComplexPrograms	AvoidTooComplexPrograms: Avoid using p high cyclomatic complexity values

Severity	Contrast rule	Engine rule ID	Description
High	Do not apply the sizeof operator to a pointer when taking the size of an array	OPT.CPP.CERTC.ARR01	ARR01: Do not apply the sizeof operator to a pointer when taking the size of an array
High	Guarantee that copies are made into storage of sufficient size	OPT.CPP.CERTC.ARR33	ARR33: Guarantee that copies are made into storage of sufficient size
High	Assumptions about the size of an environment variable	OPT.CPP.CERTC.ENV01	ENV01: Do not make assumptions about the size of an environment variable
High	Terminating Atexit handler by returning	OPT.CPP.CERTC.ENV32	ENV32: No atexit handler should terminate the program other than by returning
High	Use of sizeof() on a Pointer Type	OPT.CPP.CERTC.EXP01	EXP01: Use of sizeof() on a Pointer Type
High	Use of Uninitialized Variable	OPT.CPP.CERTC.EXP33	EXP33: Use of Uninitialized Variable
High	Functions using file names for identification	OPT.CPP.CERTC.FIO01	FIO01: Be careful using functions that use file names for identification
High	Do not assume a new-line character is read when using fgets()	OPT.CPP.CERTC.FIO36	FIO36: Do not assume a new-line character is read when using fgets()
High	Do not assume character data has been read	OPT.CPP.CERTC.FIO37	FIO37: Do not assume character data has been read
High	Check number of bits in shift operations	OPT.CPP.CERTC.INT34	INT34: In shift operations, do not shift a negative number of bits or more bits than exist in the type
High	Allocate and free memory in the same module	OPT.CPP.CERTC.MEM00	MEM00: Allocate and free memory in the same module at the same level of abstraction
High	Only Free allocated memory once	OPT.CPP.CERTC.MEM31	MEM31: Free dynamically allocated memory only once (Double Free)
High	Detect and handle memory allocation errors	OPT.CPP.CERTC.MEM32	MEM32: Detect and handle memory allocation errors
High	Race condition with link following	OPT.CPP.CERTC.POS35	POS35: Race Condition Enabling Link Following
High	Observe correct revocation order while relinquishing privileges	OPT.CPP.CERTC.POS36	POS36: Observe correct revocation order while relinquishing privileges

Severity	Contrast rule	Engine rule ID	Description
High	Improper Check for Dropped Privileges	OPT.CPP.CERTC.POS37	POS37: Improper Check for Dropped Privileges
High	Macro replacement lists should be parenthesized	OPT.CPP.CERTC.PRE02	PRE02: Macro replacement lists should be parenthesized
High	Avoid using signals to implement normal functionality	OPT.CPP.CERTC.SIG02	SIG02: Avoid using signals to implement normal functionality
High	Ensure strtok() leaves the parse string unchanged	OPT.CPP.CERTC.STR06	STR06: Do not assume that strtok() leaves string unchanged
High	Use TR 24731 for remediation of existing string manipulation	OPT.CPP.CERTC.STR07	STR07: Use TR 24731 for remediation of existing string manipulation code
High	Null-terminate byte strings as required	OPT.CPP.CERTC.STR32	STR32: Null-terminate byte strings as required
High	Do not specify the bound of a character array initialized with a string literal	OPT.CPP.CERTC.STR36	STR36: Do not specify the bound of a character array initialized with a string literal
High	Avoid Lock Unlock On Mutex	OPT.CPP.COREGL.AvoidLockUnlockOnMutex	AvoidLockUnlockOnMutex: Avoid manually locking/unlocking on mutexes, instead of using RAII
High	Call Depends On Arguments Eval Order	OPT.CPP.COREGL.CallDependsOnArgumentsEvalOrder	CallDependsOnArgumentsEvalOrder: Call functions in the evaluation order of the arguments.
High	Detached Thread	OPT.CPP.COREGL.DetachedThread	DetachedThread: Detached thread found.
High	Dont Heap Allocate Movable Result	OPT.CPP.COREGL.DontHeapAllocateMovableResult	DontHeapAllocateMovableResult: Return a stack-allocated object instead of a heap-allocated one, if it has a move constructor.
High	Generic Exception Throw	OPT.CPP.COREGL.GenericExceptionThrow	GenericExceptionThrow: Do not throw generic exceptions.
High	Move Swap Should Be No Except	OPT.CPP.COREGL.MoveSwapShouldBeNoExcept	MoveSwapShouldBeNoExcept: Move constructors should not use assignment operator and swap functions should not use noexcept.
High	Suspicious Rvalue Forward Reference	OPT.CPP.COREGL.SuspiciousRvalueForwardReference	SuspiciousRvalueForwardReference: Suspicious rvalue forwarding / rvalue reference.
High	Correct Use Memory Leaks	OPT.CPP.CorrectUseMemoryLeaks	CorrectUseMemoryLeaks: Allocated memory should be released in same scope
High	Dont Use Memory Function	OPT.CPP.DontUseMemoryFunction	DontUseMemoryFunction: Do not use malloc, realloc or free
High	Global Var Not Used Locally	OPT.CPP.GlobalVarNotUsedLocally	GlobalVarNotUsedLocally: Global variables should not be used
High	Implicit Type Conversion	OPT.CPP.ImplicitTypeConversion	ImplicitTypeConversion: Avoid function calls that require implicit type conversions

Severity	Contrast rule	Engine rule ID	Description
High	Local Vars With Global Names	OPT.CPP.LocalVarsWithGlobalNames	LocalVarsWithGlobalNames: Avoid using the same name with global and local variables
High	Avoid File Scope When Accessed From Single Function	OPT.CPP.MISRAC.AvoidFileScopeWhenAccessedFromSingleFunction	AvoidFileScopeWhenAccessedFromSingleFunction: MISRA 8.7: Objects shall be defined at block scope if they are only accessed from within a single function
High	Avoid Recursive Functions	OPT.CPP.MISRAC.AvoidRecursiveFunctions	AvoidRecursiveFunctions: MISRA 16.2: Functions shall not call themselves, either directly or indirectly
High	Do Not Check Float Equal Not Equal	OPT.CPP.MISRAC.DoNotCheckFloatEqualNotEqual	DoNotCheckFloatEqualNotEqual: MISRA 17.4: Floating-point expressions shall not be tested for equality or inequality
High	Do Not Use Dynamic Heap Allocation	OPT.CPP.MISRAC.DoNotUseDynamicHeapAllocation	DoNotUseDynamicHeapAllocation: MISRA 17.5: Dynamic heap allocation shall not be used
High	Do Not Use Reserved Name As Identifier	OPT.CPP.MISRAC.DoNotUseReservedNameAsIdentifier	DoNotUseReservedNameAsIdentifier: MISRA 17.6: Names of standard library macros, objects and functions shall not be reused
High	Do Not Use Reserved Name As Macro Name	OPT.CPP.MISRAC.DoNotUseReservedNameAsMacroName	DoNotUseReservedNameAsMacroName: MISRA 17.7: Reserved identifiers, macros and functions of the standard library shall not be defined, redefined or used
High	Do Not Use Setjmp Longjmp	OPT.CPP.MISRAC.DoNotUseSetjmpLongjmp	DoNotUseSetjmpLongjmp: MISRA 20.7: The setjmp macro and the longjmp function shall not be used
High	Do Not Use Signal Handling Functions	OPT.CPP.MISRAC.DoNotUseSignalHandlingFunctions	DoNotUseSignalHandlingFunctions: MISRA 20.8: The signal handling facilities of signal.h shall not be used
High	Do Not Use Stdio Functions	OPT.CPP.MISRAC.DoNotUseStdioFunctions	DoNotUseStdioFunctions: MISRA 20.9: The standard output library stdio.h shall not be used in production code
High	Do Not Use Time Functions	OPT.CPP.MISRAC.DoNotUseTimeFunctions	DoNotUseTimeFunctions: MISRA 20.12: The standard time handling functions of library time.h shall not be used
High	Enclose In Parentheses Macro Args	OPT.CPP.MISRAC.EncloseInParenthesesMacroArgs	EncloseInParenthesesMacroArgs: MISRA 20.13: The definition of a function-like macro each parameter shall be enclosed in parentheses
High	Explicit Type For Vars Functions	OPT.CPP.MISRAC.ExplicitTypeForVarsFunctions	ExplicitTypeForVarsFunctions: MISRA 8.2: The type of an object or function is declared or defined shall be explicitly stated
High	Function Macro Invoked With All Arguments	OPT.CPP.MISRAC.FunctionMacroInvokedWithAllArguments	FunctionMacroInvokedWithAllArguments: MISRA 17.8: A function-like macro shall not be invoked with fewer arguments
High	Identifiers Must Not Exceed 31 Chars	OPT.CPP.MISRAC.IdentifiersMustNotExceed31Chars	IdentifiersMustNotExceed31Chars: MISRA 17.9: Identifiers (internal and external) shall not have a significance of more than 31 characters
High	Initialise Auto Variables Before Use	OPT.CPP.MISRAC.InitialiseAutoVariablesBeforeUse	InitialiseAutoVariablesBeforeUse: MISRA 17.10: Automatic variables shall have been assigned a value before being used
High	Initialization For Array Structs Must Match Layout	OPT.CPP.MISRAC.InitializationForArrayStructsMustMatchLayout	InitializationForArrayStructsMustMatchLayout: MISRA 9.2: Braces shall be used to indicate and maintain the structure of the non-zero initialisation of array structures
High	Proper Bit Field Struct	OPT.CPP.MISRAC.ProperBitFieldStruct	ProperBitFieldStruct: MISRA 3.5: Bit-fields shall be of an integer type and not be mixed with non-bit-fields

Severity	Contrast rule	Engine rule ID	Description
High	Single Definition For External Linkage Identifiers	OPT.CPP.MISRAC.SingleDefinitionForExternalLinkageIdentifiers	SingleDefinitionForExternalLinkageIdentifiers: 8.9: An identifier with external linkage shall have one definition
High	Multiple Inclusion Prevention Guard	OPT.CPP.MultipleInclusionPreventionGuard	MultipleInclusionPreventionGuard: Multiple guard for headers
High	No Specify Unix Names In Include	OPT.CPP.NoSpecifyUnixNamesInInclude	NoSpecifyUnixNamesInInclude: Do not use path names in #include directives
High	Non Goto Statement	OPT.CPP.NonGotoStatement	NonGotoStatement: Do not use goto statement
High	Remove Unused Methods	OPT.CPP.RemoveUnusedMethods	RemoveUnusedMethods: Remove unused methods
High	Unspecified Parameters	OPT.CPP.UnspecifiedParameters	UnspecifiedParameters: Avoid definition of functions (variable number of parameters)
High	Avoid Multiple Inheritance	OPT.CPP.AvoidMultipleInheritance	AvoidMultipleInheritance: Avoid Classes with inheritance
High	Avoid Public Data Member	OPT.CPP.AvoidPublicDataMember	AvoidPublicDataMember: Avoid public data member
High	Dont Use Stdio Lib	OPT.CPP.DontUseStdioLib	DontUseStdioLib: Do not use the stdio.h library, use iostream.h instead
High	Law Of Big Three	OPT.CPP.LawOfBigThree	LawOfBigThree: If one of (destructor, copy constructor, copy assignment operator) is defined, the other two should be defined.
High	Remove Unused Members	OPT.CPP.RemoveUnusedMembers	RemoveUnusedMembers: Remove private members not used
High	Hardcoded Absolute Path	OPT.CPP.PORT.HardcodedAbsolutePath	HardcodedAbsolutePath: Do not hardcode absolute paths
High	Calling system() if you do not need a command processor	OPT.CPP.CERTC.ENV04	ENV04: Do not call system() if you do not need a command processor
High	Use int to capture the return value of character I/O functions	OPT.CPP.CERTC.FIO34	FIO34: Use int to capture the return value of character I/O functions
High	Temporary File created with Incorrect Permissions	OPT.CPP.CERTC.FIO43	FIO43: Creation of Temporary File in Directory with Incorrect Permissions
High	Avoid Vararg Functions	OPT.CPP.MISRAC.AvoidVarargFunctions	AvoidVarargFunctions: MISRA 16.1: Functions shall be defined with a variable number of arguments
High	Resource Injection	OPT.CPP.SEC.ResourceInjection	ResourceInjection: Improper control of resource identifiers ("Resource Injection")
High	Hardcoded Salt	OPT.CPP.SEC.HardcodedSalt	HardcodedSalt: Use of hardcoded salt
High	Insufficient Key Size	OPT.CPP.SEC.InsufficientKeySize	InsufficientKeySize: Weak cryptography, insufficient key length
High	Weak Cryptographic Hash	OPT.CPP.SEC.WeakCryptographicHash	WeakCryptographicHash: Weak cryptographic hash
High	Weak Encryption	OPT.CPP.SEC.WeakEncryption	WeakEncryption: Weak symmetric encryption
Info	Avoid Braces Same Line	OPT.CPP.AvoidBracesSameLine	AvoidBracesSameLine: Write curly brackets on separate line

Severity	Contrast rule	Engine rule ID	Description
Info	Avoid Numeric Values	OPT.CPP.AvoidNumericValues	AvoidNumericValues: Avoid numeric constants
Info	Avoid Question Mark	OPT.CPP.AvoidQuestionMark	AvoidQuestionMark: Avoid ?: ternary operator
Info	Break In Loops	OPT.CPP.BreakInLoops	BreakInLoops: Do not use break statements
Info	Avoid Explicit New Delete	OPT.CPP.COREGL.AvoidExplicitNewDelete	AvoidExplicitNewDelete: Avoid new and delete operators
Info	Class Naming Convention	OPT.CPP.ClassNamingConvention	ClassNamingConvention: Names for struct / class / namespace items must follow a naming convention
Info	Constant Naming Convention	OPT.CPP.ConstantNamingConvention	ConstantNamingConvention: Global constants must follow a naming convention
Info	Data Member Naming Convention	OPT.CPP.DataMemberNamingConvention	DataMemberNamingConvention: Data member names must follow a naming convention
Info	Forbidden Functions	OPT.CPP.ForbiddenFunctions	ForbiddenFunctions: Avoid use of discouraged functions
Info	At Most One Break In Loop	OPT.CPP.MISRAC.AtMostOneBreakInLoop	AtMostOneBreakInLoop: MISRA 14.6: For each loop statement there shall be at most one break statement used for loop termination
Info	Avoid Trigraphs	OPT.CPP.MISRAC.AvoidTrigraphs	AvoidTrigraphs: MISRA 4.2: Trigraphs shall not be used
Info	Do Not Comment Out Source Code	OPT.CPP.MISRAC.DoNotCommentOutSourceCode	DoNotCommentOutSourceCode: MISRA 2.2: No part of code should not be commented out
Info	Explicit Check Against Zero	OPT.CPP.MISRAC.ExplicitCheckAgainstZero	ExplicitCheckAgainstZero: MISRA 13.2: Tests against zero should be made explicit, unless the operand is effectively Boolean
Info	Include Not After Statements	OPT.CPP.MISRAC.IncludeNotAfterStatements	IncludeNotAfterStatements: MISRA 19.1: #include directives should only be preceeded in a file by preprocessor directives or comments
Info	Macros Naming Convention	OPT.CPP.MacrosNamingConvention	MacrosNamingConvention: Macros naming convention
Info	Maximun Line Size	OPT.CPP.MaximunLineSize	MaximunLineSize: MaxLineSize: Do not use long code lines
Info	Method Naming Convention	OPT.CPP.MethodNamingConvention	MethodNamingConvention: Functions / class methods naming convention
Info	Methods Comment Code Ratio	OPT.CPP.MethodsCommentCodeRatio	MethodsCommentCodeRatio: Avoid functions with low comment code ratio
Info	Parenthesized Functions	OPT.CPP.ParenthesizedFunctions	ParenthesizedFunctions: Write sizeof and pointer expressions in parenthesis
Info	Space Indentation	OPT.CPP.SpaceIndentation	SpaceIndentation: Allow spaces before and after operators
Info	Typedef Naming Convention	OPT.CPP.TypedefNamingConvention	TypedefNamingConvention: Names for typedef declared types must follow a naming convention
Info	Use Blocks	OPT.CPP.UseBlocks	UseBlocks: Use blocks in conditional and loop statements
Info	Use Setters	OPT.CPP.UseSetters	UseSetters: Do not perform direct instance variable assignments in constructors
Low	Avoid Dependency Cycles Between Namespaces	OPT.CPP.AvoidDependencyCyclesBetweenNamespaces	AvoidDependencyCyclesBetweenNamespaces: Avoid cyclic dependencies between namespaces

Severity	Contrast rule	Engine rule ID	Description
Low	Avoid Many Parameters Function	OPT.CPP.AvoidManyParametersFunction	AvoidManyParametersFunction: Avoid functions with too many parameters
Low	Avoid One Case Switch	OPT.CPP.AvoidOneCaseSwitch	AvoidOneCaseSwitch: Avoid switch statements with a low number of case conditions
Low	Use consistent array notation across all source files	OPT.CPP.CERTC.ARR31	ARR31: Use consistent array notation across all source files
Low	Use bitwise operators only on unsigned operands	OPT.CPP.CERTC.INT13	INT13: Use bitwise operators only on unsigned operands
Low	Do not use vfork()	OPT.CPP.CERTC.POS33	POS33: Do not use vfork()
Low	Prefer inline or static functions to function-like macros	OPT.CPP.CERTC.PRE00	PRE00: Prefer inline or static functions to function-like macros
Low	Check Names Definition And Declaration	OPT.CPP.CheckNamesDefinitionAndDeclaration	CheckNamesDefinitionAndDeclaration: For functions, check parameter names in function definition and declaration
Low	Class Comment Code Ratio	OPT.CPP.ClassCommentCodeRatio	ClassCommentCodeRatio: Avoid classes, structs, and unions with low comment/code ratio
Low	Dont Compare Pointer To Null	OPT.CPP.DontComparePointerToNull	DontComparePointerToNull: Do not compare pointers to NULL, use 0 instead
Low	Dont Compare Pointer To Zero	OPT.CPP.DontComparePointerToZero	DontComparePointerToZero: Do not compare pointers to zero, use NULL instead
Low	Including Header File	OPT.CPP.IncludingHeaderFile	IncludingHeaderFile: Avoid implementation files that include a header file with the same name
Low	Initialization Instead Assignment	OPT.CPP.InitializationInsteadAssignment	InitializationInsteadAssignment: Always use initialization instead of assignment
Low	Avoid Single Line Comments	OPT.CPP.MISRAC.AvoidSingleLineComments	AvoidSingleLineComments: MISRA 2.2: C-style single line comments (//...) shall not be used
Low	Avoid Unreachable Code	OPT.CPP.MISRAC.AvoidUnreachableCode	AvoidUnreachableCode: MISRA 14.1: The execution shall not reach unreachable code
Low	Case With Break	OPT.CPP.MISRAC.CaseWithBreak	CaseWithBreak: MISRA 15.2: An unconditional break statement shall terminate every non-empty case of a switch
Low	Comment Should Not Contain Open Comment Chars	OPT.CPP.MISRAC.CommentShouldNotContainOpenCommentChars	CommentShouldNotContainOpenCommentChars: MISRA 2.3: A comment shall not contain the characters /* or */
Low	Declare Const Pointer Param If Unchanged Value	OPT.CPP.MISRAC.DeclareConstPointerParamIfUnchangedValue	DeclareConstPointerParamIfUnchangedValue: MISRA 16.7: A pointer parameter in a function shall be declared as pointer to const if the pointer is not modified or to modify the addressed object
Low	Do Not Def Undef Macros In Blocks	OPT.CPP.MISRAC.DoNotDefUndefMacrosInBlocks	DoNotDefUndefMacrosInBlocks: MISRA 19.4: A macro shall not be defined or undefined within a block
Low	Do Not Use Atof Atoi Atol	OPT.CPP.MISRAC.DoNotUseAtofAtoiAtol	DoNotUseAtofAtoiAtol: MISRA 20.10: The functions atof, atoi and atol from library stdlib.h shall not be used
Low	Explicit Size In Extern Arrays	OPT.CPP.MISRAC.ExplicitSizeInExternArrays	ExplicitSizeInExternArrays: MISRA 8.12: A variable that is declared with external linkage, its size shall be explicitly or defined implicitly by initialization

Severity	Contrast rule	Engine rule ID	Description
Low	Function Pointer Casts	OPT.CPP.MISRAC.FunctionPointerCasts	FunctionPointerCasts: MISRA 11.1: Conversion shall not be performed between a pointer to a function and any type other than an integral type
Low	If Else If Must End With Else	OPT.CPP.MISRAC.IfElseIfMustEndWithElse	IfElseIfMustEndWithElse: MISRA 14.10: All 'if-else-if' constructs shall be terminated with an else
Low	If Else Statements Must Use Braces	OPT.CPP.MISRAC.IfElseStatementsMustUseBraces	IfElseStatementsMustUseBraces: MISRA 14.11: 'if-else' statements must use braces
Low	Logical Expression With Primary Expression Operands	OPT.CPP.MISRAC.LogicalExpressionWithPrimaryExpressionOperands	LogicalExpressionWithPrimaryExpressionOperands: MISRA 12.5: The operands of a logical && shall be primary-expressions
Low	Loops Should Use Braces	OPT.CPP.MISRAC.LoopsShouldUseBraces	LoopsShouldUseBraces: MISRA 14.8: Loop bodies shall use braces to delimit loop body
Low	Max Two Pointer Indirections	OPT.CPP.MISRAC.MaxTwoPointerIndirections	MaxTwoPointerIndirections: MISRA 17.5: Indirection objects should contain no more than 2 levels of pointer indirection
Low	No Pointer Arithmetic Except Array Index	OPT.CPP.MISRAC.NoPointerArithmeticExceptArrayIndex	NoPointerArithmeticExceptArrayIndex: MISRA 11.7: Array indexing shall be the only allowed form of pointer arithmetic
Low	No Side Effects In Right Operand Of Logical Op	OPT.CPP.MISRAC.NoSideEffectsInRightOperandOfLogicalOp	NoSideEffectsInRightOperandOfLogicalOp: MISRA 12.4: Right-hand operands of a logical && shall not contain side effects
Low	Switch Must Have Braces	OPT.CPP.MISRAC.SwitchMustHaveBraces	SwitchMustHaveBraces: MISRA 14.8: Switch statements must use braces
Low	Use Static For Internal Linkage Identifiers	OPT.CPP.MISRAC.UseStaticForInternalLinkageIdentifiers	UseStaticForInternalLinkageIdentifiers: MISRA 12.6: Use static storage specifier for definitions / declarations of objects and functions with internal linkage
Low	One Statement Per Line	OPT.CPP.OneStatementPerLine	OneStatementPerLine: Only one statement per line
Low	Only One Return	OPT.CPP.OnlyOneReturn	OnlyOneReturn: Only one 'return' statement per function
Low	Parent Class Doesnot Reference Child Classes	OPT.CPP.ParentClassDoesnotReferenceChildClasses	ParentClassDoesnotReferenceChildClasses: A base class does not reference any of its child classes
Low	Specify Return Type	OPT.CPP.SpecifyReturnType	SpecifyReturnType: Explicit specification of return type of a function
Low	Variables Never Used	OPT.CPP.VariablesNeverUsed	VariablesNeverUsed: Local variables never used
Low	Private Data Members	OPT.CPP.PrivateDataMembers	PrivateDataMembers: Max number of private data members
Low	Private Methods	OPT.CPP.PrivateMethods	PrivateMethods: Max number of private methods
Low	Protected Data Members	OPT.CPP.ProtectedDataMembers	ProtectedDataMembers: Max number of protected data members
Low	Protected Methods	OPT.CPP.ProtectedMethods	ProtectedMethods: Max number of protected methods
Low	Specify Section Order	OPT.CPP.SpecifySectionOrder	SpecifySectionOrder: In containers (class, struct, union) declare members in a certain access order
Medium	Avoid Global Vars	OPT.CPP.AvoidGlobalVars	AvoidGlobalVars: Avoid using global variables
Medium	Avoid Large Methods	OPT.CPP.AvoidLargeMethods	AvoidLargeMethods: Avoid functions and methods with too many lines of code

Severity	Contrast rule	Engine rule ID	Description
Medium	Avoid Volatile Vars	OPT.CPP.AvoidVolatileVars	AvoidVolatileVars: Do not use volatile variables
Medium	Do not form or use out-of-bounds pointers or array subscripts on arrays	OPT.CPP.CERTC.ARR30	ARR30: Do not form or use out-of-bounds array subscripts on arrays.
Medium	Allowing loops to iterate beyond the end of an array	OPT.CPP.CERTC.ARR35	ARR35: Do not allow loops to iterate beyond an array
Medium	Allowing loops to iterate beyond the end of an array	OPT.CPP.CERTC.ARR35_bis	ARR35: Do not allow loops to iterate beyond an array
Medium	Detect and handle input/output errors	OPT.CPP.CERTC.FIO33	FIO33: Detect and handle input/output errors in undefined behavior
Medium	Evaluate integer expressions	OPT.CPP.CERTC.INT35	INT35: Evaluate integer expressions in a loop before comparing or assigning to that size
Medium	Use realloc() to resize dynamically allocated arrays	OPT.CPP.CERTC.MEM08	MEM08: Use realloc() only to resize dynamically allocated arrays
Medium	Incorrect Calculation of Buffer Size	OPT.CPP.CERTC.MEM35	MEM35: Incorrect Calculation of Buffer Size
Medium	Use the readlink() function properly	OPT.CPP.CERTC.POS30	POS30: Use the readlink() function properly
Medium	Use parentheses within macros around parameter names	OPT.CPP.CERTC.PRE01	PRE01: Use parentheses within macros around parameter names
Medium	Wrap multistatement macros in a do-while loop	OPT.CPP.CERTC.PRE10	PRE10: Wrap multistatement macros in a do-while loop
Medium	Catch Exceptions By Reference	OPT.CPP.COREGL.CatchExceptionsByReference	CatchExceptionsByReference: Exceptions should always be caught by reference.
Medium	Generic Exception Catch	OPT.CPP.COREGL.GenericExceptionCatch	GenericExceptionCatch: Do not catch generic exceptions.
Medium	Polimorphic Class Should Suppress Copying	OPT.CPP.COREGL.PolimorphicClassShouldSuppressCopying	PolimorphicClassShouldSuppressCopying: Polymorphic class should suppress copying
Medium	Use Make Factories For Creating Smart Pointers	OPT.CPP.COREGL.UseMakeFactoriesForCreatingSmartPointers	UseMakeFactoriesForCreatingSmartPointers: Use factory functions for creating smart pointers
Medium	Dont Convert Const To Non Const	OPT.CPP.DontConvertConstToNonConst	DontConvertConstToNonConst: Never convert a const to a non-const

Severity	Contrast rule	Engine rule ID	Description
Medium	Include Headers Only	OPT.CPP.IncludeHeadersOnly	IncludeHeadersOnly: Avoid using #include which are not header files
Medium	All Macro Identifiers Defined Before Use	OPT.CPP.MISRAC.AllMacroIdentifiersDefinedBeforeUse	AllMacroIdentifiersDefinedBeforeUse: MISRA 12.1: macro identifiers in preprocessor directives shall be defined before use, except in #ifdef and #ifndef directives and defined() operator
Medium	Arithmetic On Pointers To Array	OPT.CPP.MISRAC.ArithmeticOnPointersToArray	ArithmeticOnPointersToArray: MISRA 17.1: arithmetic shall only be applied to pointers to an array or array element
Medium	Avoid Assignment In Boolean Expressions	OPT.CPP.MISRAC.AvoidAssignmentInBooleanExpressions	AvoidAssignmentInBooleanExpressions: MISRA 12.11: Assignment operators shall not be used in expressions that yield a boolean value
Medium	Avoid Comma Operator	OPT.CPP.MISRAC.AvoidCommaOperator	AvoidCommaOperator: MISRA 12.10: The comma operator shall not be used
Medium	Avoid Continue Statement	OPT.CPP.MISRAC.AvoidContinueStatement	AvoidContinueStatement: MISRA 14.5: The continue statement must not be used
Medium	Avoid Goto Statement	OPT.CPP.MISRAC.AvoidGotoStatement	AvoidGotoStatement: MISRA 14.4: Goto statement must not be used
Medium	Avoid Non Null Statements Without Effect	OPT.CPP.MISRAC.AvoidNonNullStatementsWithoutEffect	AvoidNonNullStatementsWithoutEffect: MISRA 12.12: non-null statements shall either have at least one side-effect however executed, or cause control flow to change
Medium	Avoid Non Standard Chars In Header Filenames	OPT.CPP.MISRAC.AvoidNonStandardCharsInHeaderFilenames	AvoidNonStandardCharsInHeaderFilenames: MISRA 19.2: Non-standard characters should not be used in header file names in #include directives
Medium	Avoid Non Standard Escape Sequences	OPT.CPP.MISRAC.AvoidNonStandardEscapeSequences	AvoidNonStandardEscapeSequences: MISRA 19.3: character constants, only those escape sequences defined in ISO C Standard shall be used
Medium	Avoid Octal Constants	OPT.CPP.MISRAC.AvoidOctalConstants	AvoidOctalConstants: MISRA 7.1: Octal constants (other than zero) and octal escape sequences shall not be used
Medium	Avoid Undef Directive	OPT.CPP.MISRAC.AvoidUndefDirective	AvoidUndefDirective: MISRA 19.6: #undef shall not be used
Medium	Avoid Unnecessary External Linkage	OPT.CPP.MISRAC.AvoidUnnecessaryExternalLinkage	AvoidUnnecessaryExternalLinkage: MISRA 12.12: declarations and definitions of objects or functions of file scope shall have internal linkage unless external linkage is required
Medium	Compare Pointers When On Same Array	OPT.CPP.MISRAC.ComparePointersWhenOnSameArray	ComparePointersWhenOnSameArray: MISRA 17.1: > [{"MISRA-C": "17.3"}]
Medium	Declare Functions At File Scope	OPT.CPP.MISRAC.DeclareFunctionsAtFileScope	DeclareFunctionsAtFileScope: MISRA 8.6: Functions shall be declared at file scope
Medium	Declare No Parameters Function As Void	OPT.CPP.MISRAC.DeclareNoParametersFunctionAsVoid	DeclareNoParametersFunctionAsVoid: MISRA 12.11: Functions with no parameters shall be declared with parameter type void
Medium	Do Not Mix Inc Dec Operators With Other Operators	OPT.CPP.MISRAC.DoNotMixIncDecOperatorsWithOtherOperators	DoNotMixIncDecOperatorsWithOtherOperators: MISRA 12.13: The increment (++) and decrement (--) operators shall not be mixed with other operators in an expression
Medium	Do Not Modify Loop Variable In Body	OPT.CPP.MISRAC.DoNotModifyLoopVariableInBody	DoNotModifyLoopVariableInBody: MISRA 12.14: Variables being used within a for loop for iteration counting shall not be modified in the loop body
Medium	Do Not Use Abort Exit Getenv System	OPT.CPP.MISRAC.DoNotUseAbortExitGetenvSystem	DoNotUseAbortExitGetenvSystem: MISRA 19.4: library functions abort, exit, getenv and system shall not be used

Severity	Contrast rule	Engine rule ID	Description
Medium	Do Not Use Errno	OPT.CPP.MISRAC.DoNotUseErrno	DoNotUseErrno: MISRA 20.5: The error in variable errno shall not be used
Medium	Do Not Use Offsetof	OPT.CPP.MISRAC.DoNotUseOffsetof	DoNotUseOffsetof: MISRA 20.6: The macro offsetof, shall not be used
Medium	Do Not Use Underlying Bit Rep Of Float	OPT.CPP.MISRAC.DoNotUseUnderlyingBitRepOfFloat	DoNotUseUnderlyingBitRepOfFloat: MISRA 20.7: Underlying bit representations of floating-point types shall not be used
Medium	Document Pragma Directives	OPT.CPP.MISRAC.DocumentPragmaDirectives	DocumentPragmaDirectives: MISRA 3.4: A #pragma directive shall be documented and its effect shall be described
Medium	Encapsulate Assembly	OPT.CPP.MISRAC.EncapsulateAssembly	EncapsulateAssembly: MISRA 2.1: Assembly code shall be encapsulated and isolated
Medium	Evaluation Order Independence	OPT.CPP.MISRAC.EvaluationOrderIndependence	EvaluationOrderIndependence: MISRA 12.1: The order of evaluation of an expression shall be the same under all circumstances that the standard permits
Medium	Float Implicit Conversions	OPT.CPP.MISRAC.FloatImplicitConversions	FloatImplicitConversions: MISRA 10.2: The value of an expression of floating type shall not be implicitly converted to a different underlying type
Medium	For Control Expression With Float Objects	OPT.CPP.MISRAC.ForControlExpressionWithFloatObjects	ForControlExpressionWithFloatObjects: MISRA 12.2: The controlling expression of a for statement shall not contain any objects of floating type
Medium	For Loop Expressions For Loop Control	OPT.CPP.MISRAC.ForLoopExpressionsForLoopControl	ForLoopExpressionsForLoopControl: MISRA 12.3: A for statement shall have three expressions of a for statement shall be limited to only with loop control
Medium	Functions Should Have Single Return At End	OPT.CPP.MISRAC.FunctionsShouldHaveSingleReturnAtEnd	FunctionsShouldHaveSingleReturnAtEnd: MISRA 11.1: Functions shall have a single point of exit at the end of the function
Medium	Identifiers Must Not Hide Outer Definitions	OPT.CPP.MISRAC.IdentifiersMustNotHideOuterDefinitions	IdentifiersMustNotHideOuterDefinitions: MISRA 2.2: Identifiers in an inner scope shall not hide outer ones by reusing the same name
Medium	Integer Implicit Conversions	OPT.CPP.MISRAC.IntegerImplicitConversions	IntegerImplicitConversions: MISRA 10.1: The value of an expression of integer type shall not be implicitly converted to a different underlying type
Medium	Macro Expansion Check	OPT.CPP.MISRAC.MacroExpansionCheck	MacroExpansionCheck: MISRA 19.4: C macro expansion shall only expand to safe constructions
Medium	Name Parameters In Function Prototypes	OPT.CPP.MISRAC.NameParametersInFunctionPrototypes	NameParametersInFunctionPrototypes: MISRA 2.3: Names shall be given for all parameters in function prototypes
Medium	Object Pointer Casts	OPT.CPP.MISRAC.ObjectPointerCasts	ObjectPointerCasts: MISRA 11.2: Conversions shall not be performed between a pointer to an object type other than an integral type, another pointer to object type, or a pointer to void
Medium	Proper Cast Complex Float Expression	OPT.CPP.MISRAC.ProperCastComplexFloatExpression	ProperCastComplexFloatExpression: MISRA 10.3: The value of a complex expression of floating type shall be cast to a floating type narrower or of the same type
Medium	Proper Cast Complex Integer Expression	OPT.CPP.MISRAC.ProperCastComplexIntegerExpression	ProperCastComplexIntegerExpression: MISRA 10.4: The value of a complex expression of integer type shall only be cast to a type of same signedness and narrower than the underlying type of the expression
Medium	Same Function Declaration And Definition	OPT.CPP.MISRAC.SameFunctionDeclarationAndDefinition	SameFunctionDeclarationAndDefinition: MISRA 2.4: For each function parameter the type in the declaration and definition shall be identical, and return type shall also be identical
Medium	Sizeof Expr With Side Effects	OPT.CPP.MISRAC.SizeofExprWithSideEffects	SizeofExprWithSideEffects: MISRA 12.3: The sizeof operator shall not be used on expressions with side effects

Severity	Contrast rule	Engine rule ID	Description
Medium	Switch With Default	OPT.CPP.MISRAC.SwitchWithDefault	SwitchWithDefault: MISRA 15.3: Avoid switch statements without a default clause
Medium	Switch Without Case Should Be Refactored	OPT.CPP.MISRAC.SwitchWithoutCaseShouldBeRefactored	SwitchWithoutCaseShouldBeRefactored: MISRA 15.3: Switch statements without any case shall be refactored
Medium	Tag Unique Identifier	OPT.CPP.MISRAC.TagUniqueIdentifier	TagUniqueIdentifier: MISRA 5.4: A tag name shall be a unique identifier
Medium	Typedef Unique Identifier	OPT.CPP.MISRAC.TypedefUniqueIdentifier	TypedefUniqueIdentifier: MISRA 5.3: A typedef name shall be a unique identifier
Medium	Unsigned Bitwise Operands	OPT.CPP.MISRAC.UnsignedBitwiseOperands	UnsignedBitwiseOperands: MISRA 12.7: Bitwise operators shall not be applied to operands of an unsigned type if the underlying type is signed
Medium	Num Max Class By Namespace	OPT.CPP.NumMaxClassByNamespace	NumMaxClassByNamespace: Avoid an excessive number of classes per package/namespace
Medium	Remove Unused Param	OPT.CPP.RemoveUnusedParam	RemoveUnusedParam: Remove unused parameters from functions
Medium	Avoid Inline Constructor And Destructor	OPT.CPP.AvoidInlineConstructorAndDestructor	AvoidInlineConstructorAndDestructor: Avoid inline constructors and destructors
Medium	Constant Member Functions	OPT.CPP.ConstantMemberFunctions	ConstantMemberFunctions: Member functions that do not modify state should be declared constant
Medium	Dont Use Cast	OPT.CPP.DontUseCast	DontUseCast: Do not use explicit type conversions (casts), excluding C++ cast operators
Medium	No Specify Member Data In Class	OPT.CPP.NoSpecifyMemberDataInClass	NoSpecifyMemberDataInClass: Never specify protected member data in a class
Medium	Non Constant Reference From Function	OPT.CPP.NonConstantReferenceFromFunction	NonConstantReferenceFromFunction: Non-constant references from public functions
Medium	Potential Infinite Loop	OPT.CPP.PotentialInfiniteLoop	PotentialInfiniteLoop: Loop with Unreachable Condition ('Infinite Loop')
Medium	Too Many Constructors	OPT.CPP.TooManyConstructors	TooManyConstructors: Avoid classes with too many constructors
Medium	Too Many Data Members	OPT.CPP.TooManyDataMembers	TooManyDataMembers: Avoid classes with too many data members
Medium	Too Many Methods	OPT.CPP.TooManyMethods	TooManyMethods: Avoid classes with too many methods
Medium	Obsolete Function	OPT.CPP.PORT.ObsoleteFunction	ObsoleteFunction: Do not use deprecated functions.
Medium	Hardcoded Username Password	OPT.CPP.SEC.HardcodedUsernamePassword	HardcodedUsernamePassword: Use of Hardcoded Credentials
Medium	Insecure Randomness	OPT.CPP.SEC.InsecureRandomness	InsecureRandomness: Standard pseudo-random number generators cannot withstand cryptographic attacks

C Scan rules

Contrast Scan supports these rules for C.

Severity	Engine rule ID	Contrast rule	Description
Critical	OPT.C.AvoidCompDiffTypes	Avoid Comp Diff Types	AvoidCompDiffTypes: Do not compare variables with different basic types

Severity	Engine rule ID	Contrast rule	Description
Critical	OPT.C.CERTC.ARR38	Adding or subtracting an integer to a pointer if resulting value does not refer to a valid array element	ARR38: Do not add or subtract an integer to a pointer if resulting value does not refer to a valid array element
Critical	OPT.C.CERTC.EXP34	NULL Pointer Dereference	EXP34: NULL Pointer Dereference
Critical	OPT.C.CERTC.MEM30	Do not access freed memory	MEM30: Do not access freed memory (Use a free)
Critical	OPT.C.CERTC.MEM34	Freeing Memory not on the Heap	MEM34: Free of Memory not on the Heap
Critical	OPT.C.CERTC.PRE09	Do not replace secure functions with less secure functions	PRE09: Do not replace secure functions with less secure functions
Critical	OPT.C.CERTC.SIG30	Signal Handler Use of a Non-reentrant Function	SIG30: Signal Handler Use of a Non-reentrant Function
Critical	OPT.C.CERTC.SIG32	Signal Handler Use of a Non-reentrant Function	SIG32: Signal Handler Use of a Non-reentrant Function
Critical	OPT.C.CERTC.STR31	Guarantee that storage for strings has sufficient space	STR31: Guarantee that storage for strings has sufficient space for character data and the null terminator
Critical	OPT.C.CERTC.STR33	Size wide character strings correctly	STR33: Size wide character strings correctly
Critical	OPT.C.CERTC.STR35	Do not copy data from an unbounded source to a fixed-length array	STR35: Do not copy data from an unbounded source to a fixed-length array
Critical	OPT.C.CheckReturnInPublicFunctions	Check Return In Public Functions	CheckReturnInPublicFunctions: Functions should return a pointer or reference to local variables
Critical	OPT.C.MISRA.NumberArgsInCallsMustMatchFormalParams	Number Args In Calls Must Match Formal Params	NumberArgsInCallsMustMatchFormalParams: MISRA 16.6: The number of arguments passed to a function shall match the number of parameters
Critical	OPT.C.SEC.AnonymousLdapBind	Anonymous Ldap Bind	AnonymousLdapBind: Access Control - Anonymous LDAP Bind
Critical	OPT.C.SEC.PathTraversal	Path Traversal	PathTraversal: Avoid non-neutralized user-controlled input composed in a pathname to access a resource
Critical	OPT.C.SEC.StaticDatabaseConnection	Static Database Connection	StaticDatabaseConnection: Static database connection / session
Critical	OPT.C.SEC.UnsafeChroot	Unsafe Chroot	UnsafeChroot: Unsafe chroot call.
Critical	OPT.C.CERTC.FIO30	Exclude unsanitized input	FIO30: Exclude unsanitized user input from file strings

Severity	Engine rule ID	Contrast rule	Description
Critical	OPT.C.CERTC.STR02	Sanitize data passed to sensitive subsystems	STR02: Sanitize data passed to sensitive subsystems
Critical	OPT.C.SEC.ConnectionStringParameterPollution	ConnectionStringParameterPollution: Connection String Parameter Pollution	ConnectionStringParameterPollution: Connection string polluted with untrusted input
Critical	OPT.C.SEC.DoSRegexp	DoS Regexp	DoSRegexp: Prevent denial of service attack through malicious regular expression ('Regex Injection')
Critical	OPT.C.SEC.LdapInjection	Ldap Injection	LdapInjection: Avoid non-neutralized user-controlled input in LDAP search filters
Critical	OPT.C.SEC.NoSQLInjection	No SQL Injection	NoSQLInjection: Improper neutralization of special elements in data query logic (NoSQL injection)
Critical	OPT.C.SEC.ProcessControl	Process Control	ProcessControl: Do not load executables or libraries from untrusted sources
Critical	OPT.C.SEC.SqlInjection	SQL Injection	SqlInjection: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
Critical	OPT.C.SEC.XmlEntityInjection	Xml Entity Injection	XmlEntityInjection: XML entity injection
Critical	OPT.C.SEC.HardcodedCryptoKey	Hardcoded Crypto Key	HardcodedCryptoKey: Hardcoded cryptographic keys
High	OPT.C.AvoidSignalManagementFunctions	Avoid Signal Management Functions	AvoidSignalManagementFunctions: Avoid using signal management functions
High	OPT.C.AvoidStructures	Avoid Structures	AvoidStructures: Avoid using certain kinds of aggregate objects (struct, union, VARIANT)
High	OPT.C.CERTC.ARR01	Do not apply the sizeof operator to a pointer when taking the size of an array	ARR01: Do not apply the sizeof operator to a pointer when taking the size of an array
High	OPT.C.CERTC.ARR33	Guarantee that copies are made into storage of sufficient size	ARR33: Guarantee that copies are made into storage of sufficient size
High	OPT.C.CERTC.ENV01	Assumptions about the size of an environment variable	ENV01: Do not make assumptions about the size of an environment variable
High	OPT.C.CERTC.ENV32	Terminating Atexit handler by returning	ENV32: No atexit handler should terminate in any way other than by returning
High	OPT.C.CERTC.EXP01	Use of sizeof() on a Pointer Type	EXP01: Use of sizeof() on a Pointer Type
High	OPT.C.CERTC.EXP33	Use of Uninitialized Variable	EXP33: Use of Uninitialized Variable
High	OPT.C.CERTC.FIO01	Functions using file names for identification	FIO01: Be careful using functions that use file names for identification
High	OPT.C.CERTC.FIO36	Do not assume a new-line character is read when using fgets()	FIO36: Do not assume a new-line character is read when using fgets()

Severity	Engine rule ID	Contrast rule	Description
High	OPT.C.CERTC.FIO37	Do not assume character data has been read	FIO37: Do not assume character data has been read
High	OPT.C.CERTC.INT34	Check number of bits in shift operations	INT34: In shift operations, do not shift a negative number of bits or more bits than exist in the left operand
High	OPT.C.CERTC.MEM00	Allocate and free memory in the same module	MEM00: Allocate and free memory in the same module at the same level of abstraction
High	OPT.C.CERTC.MEM31	Only Free allocated memory once	MEM31: Free dynamically allocated memory exactly once (Double Free)
High	OPT.C.CERTC.MEM32	Detect and handle memory allocation errors	MEM32: Detect and handle memory allocation errors
High	OPT.C.CERTC.POS35	Race condition with link following	POS35: Race Condition Enabling Link Following
High	OPT.C.CERTC.POS36	Observe correct revocation order while relinquishing privileges	POS36: Observe correct revocation order while relinquishing privileges
High	OPT.C.CERTC.POS37	Improper Check for Dropped Privileges	POS37: Improper Check for Dropped Privileges
High	OPT.C.CERTC.PRE02	Macro replacement lists should be parenthesized	PRE02: Macro replacement lists should be parenthesized
High	OPT.C.CERTC.SIG02	Avoid using signals to implement normal functionality	SIG02: Avoid using signals to implement normal functionality
High	OPT.C.CERTC.STR06	Ensure strtok() leaves the parse string unchanged	STR06: Do not assume that strtok() leaves the parse string unchanged
High	OPT.C.CERTC.STR07	Use TR 24731 for remediation of existing string manipulation	STR07: Use TR 24731 for remediation of existing string manipulation code
High	OPT.C.CERTC.STR32	Null-terminate byte strings as required	STR32: Null-terminate byte strings as required
High	OPT.C.CERTC.STR36	Do not specify the bound of a character array initialized with a string literal	STR36: Do not specify the bound of a character array initialized with a string literal
High	OPT.C.CorrectUseMemoryLeaks	Correct Use Memory Leaks	CorrectUseMemoryLeaks: Allocated memory be released in same scope
High	OPT.C.DontUseMemoryFunction	Dont Use Memory Function	DontUseMemoryFunction: Do not use malloc, calloc, realloc or free

Severity	Engine rule ID	Contrast rule	Description
High	OPT.C.GlobalVarNotUsedLocally	Global Var Not Used Locally	GlobalVarNotUsedLocally: Global variables not locally used
High	OPT.C.ImplicitTypeConversion	Implicit Type Conversion	ImplicitTypeConversion: Avoid function calls that cause implicit type conversions
High	OPT.C.LocalVarsWithGlobalNames	Local Vars With Global Names	LocalVarsWithGlobalNames: Avoid using the name with global and local variables
High	OPT.C.MISRAC.AvoidFileScopeWhenAccessedFromSingleFunction	Avoid File Scope When Accessed From Single Function	AvoidFileScopeWhenAccessedFromSingleFunction: MISRA 8.7: Objects shall be defined at block level if they are only accessed from within a single function
High	OPT.C.MISRAC.AvoidRecursiveFunctions	Avoid Recursive Functions	AvoidRecursiveFunctions: MISRA 16.2: Functions shall not call themselves, either directly or indirectly
High	OPT.C.MISRAC.DoNotCheckFloatEqualNotEqual	Do Not Check Float Equal Not Equal	DoNotCheckFloatEqualNotEqual: MISRA 13.9: Floating-point expressions shall not be tested for equality or inequality
High	OPT.C.MISRAC.DoNotUseDynamicHeapAllocation	Do Not Use Dynamic Heap Allocation	DoNotUseDynamicHeapAllocation: MISRA 20.1: Dynamic heap allocation shall not be used
High	OPT.C.MISRAC.DoNotUseReservedNameAsIdentifier	Do Not Use Reserved Name As Identifier	DoNotUseReservedNameAsIdentifier: MISRA 20.1: The names of standard library macros, objects, and functions shall not be reused
High	OPT.C.MISRAC.DoNotUseReservedNameAsMacroName	Do Not Use Reserved Name As Macro Name	DoNotUseReservedNameAsMacroName: MISRA 20.1: Reserved identifiers, macros and functions in the standard library shall not be defined, redefined, or undefined
High	OPT.C.MISRAC.DoNotUseSetjmpLongjmp	Do Not Use Setjmp Longjmp	DoNotUseSetjmpLongjmp: MISRA 20.7: The setjmp macro and the longjmp function shall not be used
High	OPT.C.MISRAC.DoNotUseSignalHandlingFunctions	Do Not Use Signal Handling Functions	DoNotUseSignalHandlingFunctions: MISRA 20.8: The signal handling facilities of signal.h shall not be used
High	OPT.C.MISRAC.DoNotUseStdioFunctions	Do Not Use Stdio Functions	DoNotUseStdioFunctions: MISRA 20.9: The input and output library stdio.h shall not be used in production code
High	OPT.C.MISRAC.DoNotUseTimeFunctions	Do Not Use Time Functions	DoNotUseTimeFunctions: MISRA 20.12: The time handling functions of library time.h shall not be used
High	OPT.C.MISRAC.EncloseInParanthesesMacroArgs	Enclose In Parantheses Macro Args	EncloseInParanthesesMacroArgs: MISRA 19.1: In the definition of a function-like macro each parameter shall be enclosed in parentheses
High	OPT.C.MISRAC.ExplicitTypeForVarsFunctions	Explicit Type For Vars Functions	ExplicitTypeForVarsFunctions: MISRA 8.2: Whenever an object or function is declared or defined, its type shall be explicitly stated
High	OPT.C.MISRAC.FunctionMacroInvokedWithAllArguments	Function Macro Invoked With All Arguments	FunctionMacroInvokedWithAllArguments: MISRA 19.8: A function-like macro shall not be invoked without all of its arguments
High	OPT.C.MISRAC.IdentifiersMustNotExceed31Chars	Identifiers Must Not Exceed 31 Chars	IdentifiersMustNotExceed31Chars: MISRA 5.1: Identifiers (internal and external) shall not rely on the significance of more than 31 characters
High	OPT.C.MISRAC.InitialiseAutoVariablesBeforeUse	Initialise Auto Variables Before Use	InitialiseAutoVariablesBeforeUse: MISRA 9.1: Automatic variables shall have been assigned a value before being used
High	OPT.C.MISRAC.InitializationForArrayStructsMustMatchLayout	Initialization For Array Structs Must Match Layout	InitializationForArrayStructsMustMatchLayout: MISRA 9.2: Braces shall be used to indicate a match the structure of the non-zero initialisation arrays and structures

Severity	Engine rule ID	Contrast rule	Description
High	OPT.C.MISRAC.ProperBitFieldsStruct	Proper Bit Field Struct	ProperBitFieldsStruct: MISRA 3.5: Bit-fields in should use int type and not be mixed with non fields
High	OPT.C.MISRAC.SingleDefinitionForExternalLinkageIdentifiers	Single Definition For External Linkage Identifiers	SingleDefinitionForExternalLinkageIdentifiers: MISRA 8.9: An identifier with external linkage have exactly one definition
High	OPT.C.MultipleInclusionPreventionGuard	Multiple Inclusion Prevention Guard	MultipleInclusionPreventionGuard: Multiple inclusion guard for headers
High	OPT.C.NoSpecifyUnixNamesInInclude	No Specify Unix Names In Include	NoSpecifyUnixNamesInInclude: Do not use absolute path names in #include directives
High	OPT.C.NonGotoStatement	Non Goto Statement	NonGotoStatement: Do not use goto statement
High	OPT.C.RemoveUnusedMethods	Remove Unused Methods	RemoveUnusedMethods: Remove unused functions
High	OPT.C.UnspecifiedParameters	Unspecified Parameters	UnspecifiedParameters: Avoid definition of var functions (variable number of parameters)
High	OPT.C.PORT.HardcodedAbsolutePath	Hardcoded Absolute Path	HardcodedAbsolutePath: Do not hardcode absolute paths
High	OPT.C.CERTC.ENV04	Calling system() if you do not need a command processor	ENV04: Do not call system() if you do not need a command processor
High	OPT.C.CERTC.FIO34	Use int to capture the return value of character I/O functions	FIO34: Use int to capture the return value of character I/O functions
High	OPT.C.CERTC.FIO43	Temporary File created with Incorrect Permissions	FIO43: Creation of Temporary File in Directory with Incorrect Permissions
High	OPT.C.MISRAC.AvoidVarargFunctions	Avoid Vararg Functions	AvoidVarargFunctions: MISRA 16.1: Function prototype not be defined with a variable number of arguments
High	OPT.C.SEC.ResourceInjection	Resource Injection	ResourceInjection: Improper control of resource identifiers ("Resource Injection")
High	OPT.C.SEC.HardcodedSalt	Hardcoded Salt	HardcodedSalt: Use of hardcoded salt
High	OPT.C.SEC.InsufficientKeySize	Insufficient Key Size	InsufficientKeySize: Weak cryptography, insufficient key length
High	OPT.C.SEC.WeakCryptographicHash	Weak Cryptographic Hash	WeakCryptographicHash: Weak cryptographic hash
High	OPT.C.SEC.WeakEncryption	Weak Encryption	WeakEncryption: Weak symmetric encryption algorithm
Info	OPT.C.AvoidBracesSameLine	Avoid Braces Same Line	AvoidBracesSameLine: Write curly brackets { on separate line
Info	OPT.C.AvoidNumericValues	Avoid Numeric Values	AvoidNumericValues: Avoid numeric constant in code
Info	OPT.C.AvoidQuestionMark	Avoid Question Mark	AvoidQuestionMark: Avoid ?: ternary operator
Info	OPT.C.BreakInLoops	Break In Loops	BreakInLoops: Do not use break statement in loops
Info	OPT.C.ClassNamingConvention	Class Naming Convention	ClassNamingConvention: Names for struct / union / class / namespace items must follow a naming convention

Severity	Engine rule ID	Contrast rule	Description
Info	OPT.C.ConstantNamingConvention	Constant Naming Convention	ConstantNamingConvention: Global constant naming convention
Info	OPT.C.DataMemberNamingConvention	Data Member Naming Convention	DataMemberNamingConvention: Data member naming convention
Info	OPT.C.ForbiddenFunctions	Forbidden Functions	ForbiddenFunctions: Avoid use of discouraged functions
Info	OPT.C.MISRAC.AtMostOneBreakInLoop	At Most One Break In Loop	AtMostOneBreakInLoop: MISRA 14.6: For any iteration statement there shall be at most one statement used for loop termination
Info	OPT.C.MISRAC.AvoidTrigraphs	Avoid Trigraphs	AvoidTrigraphs: MISRA 4.2: Trigraphs shall not be used
Info	OPT.C.MISRAC.DoNotCommentOutSourceCode	Do Not Comment Out Source Code	DoNotCommentOutSourceCode: MISRA 2.4: Sections of code should not be commented out
Info	OPT.C.MISRAC.ExplicitCheckAgainstZero	Explicit Check Against Zero	ExplicitCheckAgainstZero: MISRA 13.2: Tests for a value against zero should be made explicit, unless the operand is effectively Boolean
Info	OPT.C.MISRAC.IncludeNotAfterStatements	Include Not After Statements	IncludeNotAfterStatements: MISRA 19.1: #include directives should only be preceded in a file by other preprocessor directives or comments
Info	OPT.C.MacrosNamingConvention	Macros Naming Convention	MacrosNamingConvention: Macros naming convention
Info	OPT.C.MaximunLineSize	Maximun Line Size	MaximunLineSize: MaxLineSize: Do not use long code lines
Info	OPT.C.MethodNamingConvention	Method Naming Convention	MethodNamingConvention: Functions / class methods naming convention
Info	OPT.C.MethodsCommentCodeRatio	Methods Comment Code Ratio	MethodsCommentCodeRatio: Avoid functions with low comment code ratio
Info	OPT.C.ParenthesizedFunctions	Parenthesized Functions	ParenthesizedFunctions: Write sizeof and return with parenthesis
Info	OPT.C.SpaceIndentation	Space Indentation	SpaceIndentation: Allow spaces before and after operators
Info	OPT.C.TypedefNamingConvention	Typedef Naming Convention	TypedefNamingConvention: Names for typedef declared types must follow a naming convention
Info	OPT.C.UseBlocks	Use Blocks	UseBlocks: Use blocks in conditional and iteration statements
Low	OPT.C.AvoidManyParametersFunction	Avoid Many Parameters Function	AvoidManyParametersFunction: Avoid functions with too many parameters
Low	OPT.C.AvoidOneCaseSwitch	Avoid One Case Switch	AvoidOneCaseSwitch: Avoid switch statements with a low number of case conditions
Low	OPT.C.CERTC.ARR31	Use consistent array notation across all source files	ARR31: Use consistent array notation across all source files
Low	OPT.C.CERTC.INT13	Use bitwise operators only on unsigned operands	INT13: Use bitwise operators only on unsigned operands
Low	OPT.C.CERTC.POS33	Do not use vfork()	POS33: Do not use vfork()
Low	OPT.C.CERTC.PRE00	Prefer inline or static functions to function-like macros	PRE00: Prefer inline or static functions to function-like macros

Severity	Engine rule ID	Contrast rule	Description
Low	OPT.C.CheckNamesDefinitionAndDeclaration	Check Names Definition And Declaration	CheckNamesDefinitionAndDeclaration: Form parameters names in function definition and declaration
Low	OPT.C.ClassCommentCodeRatio	Class Comment Code Ratio	ClassCommentCodeRatio: Avoid classes, structures, unions with low comment/code ratio
Low	OPT.C.DontComparePointerToNull	Dont Compare Pointer To Null	DontComparePointerToNull: Do not compare pointer to NULL, use 0 instead
Low	OPT.C.DontComparePointerToZero	Dont Compare Pointer To Zero	DontComparePointerToZero: Do not compare pointer to zero, use NULL instead
Low	OPT.C.IncludingHeaderFile	Including Header File	IncludingHeaderFile: Avoid implementation files do not include a header file with the same name
Low	OPT.C.InitializationInsteadAssignment	Initialization Instead Assignment	InitializationInsteadAssignment: Always use initialization instead of assignment
Low	OPT.C.MISRAC.AvoidSingleLineComments	Avoid Single Line Comments	AvoidSingleLineComments: MISRA 2.2: C99 single line comments (<code>//...</code>) shall not be used
Low	OPT.C.MISRAC.AvoidUnreachableCode	Avoid Unreachable Code	AvoidUnreachableCode: MISRA 14.1: There shall be no unreachable code
Low	OPT.C.MISRAC.CaseWithBreak	Case With Break	CaseWithBreak: MISRA 15.2: An unconditional break statement shall terminate every non-empty case clause of a switch
Low	OPT.C.MISRAC.CommentShouldNotContainOpenCommentChars	Comment Should Not Contain Open Comment Chars	CommentShouldNotContainOpenCommentChars: MISRA 2.3: A comment shall not contain the string <code>/*</code>
Low	OPT.C.MISRAC.DeclareConstPointerParamIfUnchangedValue	Declare Const Pointer Param If Unchanged Value	DeclareConstPointerParamIfUnchangedValue: MISRA 16.7: A pointer parameter in a function should be declared as pointer to const if the pointer is not used to modify the addressed object
Low	OPT.C.MISRAC.DoNotDefUndefMacrosInBlocks	Do Not Def Undef Macros In Blocks	DoNotDefUndefMacrosInBlocks: MISRA 19.5: Macros shall not be defined or undefined within a block
Low	OPT.C.MISRAC.DoNotUseAtofAtoiAtol	Do Not Use Atof Atoi Atol	DoNotUseAtofAtoiAtol: MISRA 20.10: The library functions <code>atof</code> , <code>atoi</code> and <code>atol</code> from library <code>stdlib.h</code> shall not be used
Low	OPT.C.MISRAC.ExplicitSizeInExternArrays	Explicit Size In Extern Arrays	ExplicitSizeInExternArrays: MISRA 8.12: When an array is declared with external linkage, its size shall be stated explicitly or defined implicitly by a static initialization
Low	OPT.C.MISRAC.FunctionPointerCasts	Function Pointer Casts	FunctionPointerCasts: MISRA 11.1: Conversions shall not be performed between a pointer to a function and any type other than an integral type
Low	OPT.C.MISRAC.IfElseIfMustEndWithElse	If Else If Must End With Else	IfElseIfMustEndWithElse: MISRA 14.10: All if-else-if constructs shall be terminated with an else clause
Low	OPT.C.MISRAC.IfElseStatementsMustUseBraces	If Else Statements Must Use Braces	IfElseStatementsMustUseBraces: MISRA 14.11: if-else statements must use braces
Low	OPT.C.MISRAC.LogicalExpressionWithPrimaryExpressionOperands	Logical Expression With Primary Expression Operands	LogicalExpressionWithPrimaryExpressionOperands: MISRA 12.5: The operands of a logical <code>&&</code> and <code> </code> shall be primary-expressions
Low	OPT.C.MISRAC.LoopsShouldUseBraces	Loops Should Use Braces	LoopsShouldUseBraces: MISRA 14.8: Loops shall use braces to delimit loop body
Low	OPT.C.MISRAC.MaxTwoPointerIndirections	Max Two Pointer Indirections	MaxTwoPointerIndirections: MISRA 17.5: Declaration of objects should contain no more than 2 levels of pointer indirection

Severity	Engine rule ID	Contrast rule	Description
Low	OPT.C.MISRAC.NoPointerArithmeticExceptArrayIndex	No Pointer Arithmetic Except Array Index	NoPointerArithmeticExceptArrayIndex: MISRA 12.4: Array indexing shall be the only allowed form of pointer arithmetic
Low	OPT.C.MISRAC.NoSideEffectsInRightOperandOfLogicalOp	No Side Effects In Right Operand Of Logical Op	NoSideEffectsInRightOperandOfLogicalOp: MISRA 12.4: Right-hand operands of a logical && or operator shall not contain side effects
Low	OPT.C.MISRAC.SwitchMustHaveBraces	Switch Must Have Braces	SwitchMustHaveBraces: MISRA 14.8: Switch statements must use braces
Low	OPT.C.MISRAC.UseStaticForInternalLinkageIdentifiers	Use Static For Internal Linkage Identifiers	UseStaticForInternalLinkageIdentifiers: MISRA 12.3: Use static storage specifier for definitions / declarations of objects and functions with internal linkage
Low	OPT.C.OneStatementPerLine	One Statement Per Line	OneStatementPerLine: Only one statement per line
Low	OPT.C.OnlyOneReturn	Only One Return	OnlyOneReturn: Only one 'return' statement per function
Low	OPT.C.SpecifyReturnType	Specify Return Type	SpecifyReturnType: Explicit specification of the return type of a function
Low	OPT.C.VariablesNeverUsed	Variables Never Used	VariablesNeverUsed: Local variables never used
Medium	OPT.C.AvoidGlobalVars	Avoid Global Vars	AvoidGlobalVars: Avoid using global variables
Medium	OPT.C.AvoidLargeMethods	Avoid Large Methods	AvoidLargeMethods: Avoid functions and methods with too many lines of code
Medium	OPT.C.AvoidVolatileVars	Avoid Volatile Vars	AvoidVolatileVars: Do not use volatile variables
Medium	OPT.C.CERTC.ARR30	Do not form or use out-of-bounds pointers or array subscripts on arrays	ARR30: Do not form or use out-of-bounds pointers or array subscripts on arrays.
Medium	OPT.C.CERTC.ARR35	Allowing loops to iterate beyond the end of an array	ARR35: Do not allow loops to iterate beyond the end of an array
Medium	OPT.C.CERTC.ARR35_bis	Allowing loops to iterate beyond the end of an array	ARR35: Do not allow loops to iterate beyond the end of an array
Medium	OPT.C.CERTC.FIO33	Detect and handle input/output errors	FIO33: Detect and handle input/output errors resulting in undefined behavior
Medium	OPT.C.CERTC.INT35	Evaluate integer expressions	INT35: Evaluate integer expressions in a larger size before comparing or assigning to that size
Medium	OPT.C.CERTC.MEM08	Use realloc() to resize dynamically allocated arrays	MEM08: Use realloc() only to resize dynamically allocated arrays
Medium	OPT.C.CERTC.MEM35	Incorrect Calculation of Buffer Size	MEM35: Incorrect Calculation of Buffer Size.
Medium	OPT.C.CERTC.POS30	Use the readlink() function properly	POS30: Use the readlink() function properly

Severity	Engine rule ID	Contrast rule	Description
Medium	OPT.C.CERTC.PRE01	Use parentheses within macros around parameter names	PRE01: Use parentheses within macros around parameter names
Medium	OPT.C.CERTC.PRE10	Wrap multistatement macros in a do-while loop	PRE10: Wrap multistatement macros in a do-loop
Medium	OPT.C.DontConvertConstToNonConst	Dont Convert Const To Non Const	DontConvertConstToNonConst: Never convert const to a non-const
Medium	OPT.C.IncludeHeadersOnly	Include Headers Only	IncludeHeadersOnly: Avoid using #include without which are not header files
Medium	OPT.C.MISRAC.AllMacroIdentifiersDefinedBeforeUse	All Macro Identifiers Defined Before Use	AllMacroIdentifiersDefinedBeforeUse: MISRA C89 7.1: All macro identifiers in preprocessor directives shall be defined before use, except in #ifdef and #ifndef directives and defined() operator
Medium	OPT.C.MISRAC.ArithmeticOnPointersToArray	Arithmetic On Pointers To Array	ArithmeticOnPointersToArray: MISRA 17.1: Pointer arithmetic shall only be applied to pointers that address an array or array element
Medium	OPT.C.MISRAC.AvoidAssignmentInBooleanExpressions	Avoid Assignment In Boolean Expressions	AvoidAssignmentInBooleanExpressions: MISRA 13.1: Assignment operators shall not be used in expressions that yield a boolean value
Medium	OPT.C.MISRAC.AvoidCommaOperator	Avoid Comma Operator	AvoidCommaOperator: MISRA 12.10: The comma operator shall not be used
Medium	OPT.C.MISRAC.AvoidContinueStatement	Avoid Continue Statement	AvoidContinueStatement: MISRA 14.5: continue statement must not be used
Medium	OPT.C.MISRAC.AvoidGotoStatement	Avoid Goto Statement	AvoidGotoStatement: MISRA 14.4: Goto statements must not be used
Medium	OPT.C.MISRAC.AvoidNonNullStatementsWithoutEffect	Avoid Non Null Statements Without Effect	AvoidNonNullStatementsWithoutEffect: MISRA 14.2: All non-null statements shall either have at least one side-effect however executed, or cause control flow to change
Medium	OPT.C.MISRAC.AvoidNonStandardCharsInHeaderFileNames	Avoid Non Standard Chars In Header FileNames	AvoidNonStandardCharsInHeaderFileNames: MISRA 19.2: Non-standard characters should not occur in header file names in #include directives
Medium	OPT.C.MISRAC.AvoidNonStandardEscapeSequences	Avoid Non Standard Escape Sequences	AvoidNonStandardEscapeSequences: MISRA 19.3: For character constants, only those escape sequences defined in ISO C Standard shall be used
Medium	OPT.C.MISRAC.AvoidOctalConstants	Avoid Octal Constants	AvoidOctalConstants: MISRA 7.1: Octal constants (other than zero) and octal escape sequences shall not be used
Medium	OPT.C.MISRAC.AvoidUndefDirective	Avoid Undef Directive	AvoidUndefDirective: MISRA 19.6: #undef shall not be used
Medium	OPT.C.MISRAC.AvoidUnnecessaryExternalLinkage	Avoid Unnecessary External Linkage	AvoidUnnecessaryExternalLinkage: MISRA 8.1: declarations and definitions of objects or functions at file scope shall have internal linkage unless external linkage is required
Medium	OPT.C.MISRAC.ComparePointersWhenOnSameArray	Compare Pointers When On Same Array	ComparePointersWhenOnSameArray: MISRA 17.3: >, >=, <, <=, !=, ==, [], [], [], [] MISRA-C': ['17.3']
Medium	OPT.C.MISRAC.DeclareFunctionsAtFileScope	Declare Functions At File Scope	DeclareFunctionsAtFileScope: MISRA 8.6: Functions shall be declared at file scope

Severity	Engine rule ID	Contrast rule	Description
Medium	OPT.C.MISRAC.DeclareNoParametersFunctionAsVoid	Declare No Parameters Function As Void	DeclareNoParametersFunctionAsVoid: MISRA 2.2: Functions with no parameters shall be declared with parameter type void
Medium	OPT.C.MISRAC.DoNotMixIncDecOperatorsWithOtherOperators	Do Not Mix Inc Dec Operators With Other Operators	DoNotMixIncDecOperatorsWithOtherOperators: MISRA 12.13: The increment (++) and decrement (--) operators shall not be mixed with other operators in an expression
Medium	OPT.C.MISRAC.DoNotModifyLoopVariableInBody	Do Not Modify Loop Variable In Body	DoNotModifyLoopVariableInBody: MISRA 13.3: Variables being used within a for loop for iteration counting shall not be modified in the loop body
Medium	OPT.C.MISRAC.DoNotUseAbortExitGetenvSystem	Do Not Use Abort Exit Getenv System	DoNotUseAbortExitGetenvSystem: MISRA 20.4: The library functions abort, exit, getenv and system from library stdlib.h shall not be used
Medium	OPT.C.MISRAC.DoNotUseErrno	Do Not Use Errno	DoNotUseErrno: MISRA 20.5: The error indicator variable errno shall not be used
Medium	OPT.C.MISRAC.DoNotUseOffsetof	Do Not Use Offsetof	DoNotUseOffsetof: MISRA 20.6: The macro offsetof, in stddef.h, shall not be used
Medium	OPT.C.MISRAC.DoNotUseUnderlyingBitRepOfFloat	Do Not Use Underlying Bit Rep Of Float	DoNotUseUnderlyingBitRepOfFloat: MISRA 11.6: Underlying bit representations of floating-point values shall not be used
Medium	OPT.C.MISRAC.DocumentPragmaDirectives	Document Pragma Directives	DocumentPragmaDirectives: MISRA 3.4: All uses of the #pragma directive shall be documented and explained
Medium	OPT.C.MISRAC.EncapsulateAssembly	Encapsulate Assembly	EncapsulateAssembly: MISRA 2.1: Assembly language shall be encapsulated and isolated
Medium	OPT.C.MISRAC.EvaluationOrderIndependence	Evaluation Order Independence	EvaluationOrderIndependence: MISRA 12.2: The value of an expression shall be the same under the order of evaluation that the standard permits
Medium	OPT.C.MISRAC.FloatImplicitConversions	Float Implicit Conversions	FloatImplicitConversions: MISRA 10.2: The value of an expression of floating type shall not be implicitly converted to a different underlying type
Medium	OPT.C.MISRAC.ForControlExpressionWithFloatObjects	For Control Expression With Float Objects	ForControlExpressionWithFloatObjects: MISRA 13.4: The controlling expression of a for statement shall not contain any objects of floating type
Medium	OPT.C.MISRAC.ForLoopExpressionsForLoopControl	For Loop Expressions For Loop Control	ForLoopExpressionsForLoopControl: MISRA 16.2: The three expressions of a for statement shall be concerned only with loop control
Medium	OPT.C.MISRAC.FunctionsShouldHaveSingleReturnAtEnd	Functions Should Have Single Return At End	FunctionsShouldHaveSingleReturnAtEnd: MISRA 14.7: Functions shall have a single point of exit at the end of the function
Medium	OPT.C.MISRAC.IdentifiersMustNotHideOuterDefinitions	Identifiers Must Not Hide Outer Definitions	IdentifiersMustNotHideOuterDefinitions: MISRA 23.2: Identifiers in an inner scope shall not hide outer scope ones by reusing the same name
Medium	OPT.C.MISRAC.IntegerImplicitConversions	Integer Implicit Conversions	IntegerImplicitConversions: MISRA 10.1: The value of an expression of integer type shall not be implicitly converted to a different underlying type
Medium	OPT.C.MISRAC.MacroExpansionCheck	Macro Expansion Check	MacroExpansionCheck: MISRA 19.4: C macro expansion shall only expand to safe constructions
Medium	OPT.C.MISRAC.NameParametersInFunctionPrototypes	Name Parameters In Function Prototypes	NameParametersInFunctionPrototypes: MISRA 16.3: Names shall be given for all parameters in function prototype
Medium	OPT.C.MISRAC.ObjectPointerCasts	Object Pointer Casts	ObjectPointerCasts: MISRA 11.2: Conversion shall not be performed between a pointer to an object type and any type other than an integral type, another pointer to object type, or a pointer to void

Severity	Engine rule ID	Contrast rule	Description
Medium	OPT.C.MISRAC.ProperCastComplexFloatExpression	Proper Cast Complex Float Expression	ProperCastComplexFloatExpression: MISRA 10.3: The value of a complex expression of floating point type shall only be cast to a floating type narrower than the original type and of the same size
Medium	OPT.C.MISRAC.ProperCastComplexIntegerExpression	Proper Cast Complex Integer Expression	ProperCastComplexIntegerExpression: MISRA 10.3: The value of a complex expression of integer type shall only be cast to a type of signedness no wider than the underlying type of the expression
Medium	OPT.C.MISRAC.SameFunctionDeclarationAndDefinition	Same Function Declaration And Definition	SameFunctionDeclarationAndDefinition: MISRA 8.3: For each function parameter the type in the declaration and definition shall be identical, and the return types shall also be identical
Medium	OPT.C.MISRAC.SizeofExprWithSideEffects	Sizeof Expr With Side Effects	SizeofExprWithSideEffects: MISRA 12.3: The sizeof operator shall not be used on expressions that contain side effects
Medium	OPT.C.MISRAC.SwitchWithDefault	Switch With Default	SwitchWithDefault: MISRA 15.3: Avoid switch statements without a default clause
Medium	OPT.C.MISRAC.SwitchWithoutCaseShouldBeRefactored	Switch Without Case Should Be Refactored	SwitchWithoutCaseShouldBeRefactored: MISRA 15.5: Switch statements without any case shall be refactored
Medium	OPT.C.MISRAC.TagUniqueIdentifier	Tag Unique Identifier	TagUniqueIdentifier: MISRA 5.4: A tag name shall be a unique identifier
Medium	OPT.C.MISRAC.TypedefUniqueIdentifier	Typedef Unique Identifier	TypedefUniqueIdentifier: MISRA 5.3: A typedef name shall be a unique identifier
Medium	OPT.C.MISRAC.UnsignedBitwiseOperands	Unsigned Bitwise Operands	UnsignedBitwiseOperands: MISRA 12.7: Bitwise operators shall not be applied to operands whose underlying type is signed
Medium	OPT.C.PotentialInfiniteLoop	Potential Infinite Loop	PotentialInfiniteLoop: Loop with Unreachable Condition ('Infinite Loop')
Medium	OPT.C.RemoveUnusedParam	Remove Unused Param	RemoveUnusedParam: Remove unused parameters in functions
Medium	OPT.C.PORT.ObsoleteFunction	Obsolete Function	ObsoleteFunction: Do not use deprecated or obsolete functions.
Medium	OPT.C.SEC.HardcodedUsernamePassword	Hardcoded Username Password	HardcodedUsernamePassword: Use of Hard-coded Credentials
Medium	OPT.C.SEC.InsecureRandomness	Insecure Randomness	InsecureRandomness: Standard pseudo-random number generators cannot withstand cryptographic attacks

Go Scan rules

Contrast Scan supports these rules for Go.

Severity	Contrast rule	Engine rule ID	Description
Critical	Insecure SSL	OPT.GO.SECURITY.InsecureSSL	InsecureSSL: Insecure SSL configuration
Critical	Too Much Origins Allowed	OPT.GO.SECURITY.TooMuchOriginsAllowed	TooMuchOriginsAllowed: CORS policy (Cross-origin resource sharing) too broad
Critical	Anonymous Ldap Bind	OPT.GO.SECURITY.AnonymousLdapBind	AnonymousLdapBind: Access Control - Anonymous LDAP Bind
Critical	Forbidden Call	OPT.GO.SECURITY.ForbiddenCall	ForbiddenCall: Dangerous function called.
Critical	Code Injection	OPT.GO.SECURITY.CodeInjection	CodeInjection: Avoid non-neutralized user-controlled input in dynamic code evaluation

Severity	Contrast rule	Engine rule ID	Description
Critical	Command Injection	OPT.GO.SECURITY.CommandInjection	CommandInjection: Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
Critical	Connection String Parameter Pollution	OPT.GO.SECURITY.ConnectionStringParameterPollution	ConnectionStringParameterPollution: Connection string polluted with untrusted input
Critical	Cross Site Scripting	OPT.GO.SECURITY.CrossSiteScripting	CrossSiteScripting: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
Critical	Http Splitting	OPT.GO.SECURITY.HttpSplitting	HttpSplitting: Improper Neutralization of CRLF Sequences in HTTP Headers ('HTTP Response Splitting')
Critical	Ldap Injection	OPT.GO.SECURITY.LdapInjection	LdapInjection: Avoid non-neutralized user-controlled input in LDAP search filters
Critical	Mail Command Injection	OPT.GO.SECURITY.MailCommandInjection	MailCommandInjection: Mail Command Injection
Critical	No SQL Injection	OPT.GO.SECURITY.NoSQLInjection	NoSQLInjection: Improper neutralization of special elements in data query logic (NoSQL injection)
Critical	Process Control	OPT.GO.SECURITY.ProcessControl	ProcessControl: Do not load executables or libraries from untrusted sources
Critical	Regex Injection	OPT.GO.SECURITY.RegexInjection	RegexInjection: Prevent denial of service attack through malicious regular expression ('Regex Injection')
Critical	Same Origin Method Execution	OPT.GO.SECURITY.SameOriginMethodExecution	SameOriginMethodExecution: Same Origin Method Execution (SOME)
Critical	Path Traversal	OPT.GO.SECURITY.PathTraversal	PathTraversal: Avoid non-neutralized user-controlled input composed in a pathname to a resource
Critical	SQL Injection	OPT.GO.SECURITY.SqlInjection	SqlInjection: Avoid SQL code formed with non neutralized user input (vulnerable to SQL Injection attacks)
Critical	XPath Injection	OPT.GO.SECURITY.XPathInjection	XPathInjection: Avoid XPath expressions formed with non neutralized user input
Critical	Password In Redirect	OPT.GO.SECURITY.PasswordInRedirect	PasswordInRedirect: Password Management - Password in Redirect
Critical	Hardcoded Crypto Key	OPT.GO.SECURITY.HardcodedCryptoKey	HardcodedCryptoKey: Hardcoded cryptographic keys
Critical	Non Random IV With CBC Mode	OPT.GO.SECURITY.NonRandomIVWithCBCMode	NonRandomIVWithCBCMode: Not using a Random IV with CBC Mode
Critical	Weak Cryptographic Hash	OPT.GO.SECURITY.WeakCryptographicHash	WeakCryptographicHash: Weak cryptographic hashes cannot guarantee data integrity
Critical	Weak Encryption	OPT.GO.SECURITY.WeakEncryption	WeakEncryption: Weak symmetric encryption algorithm
High	Insufficient Session Expiration	OPT.GO.SECURITY.InsufficientSessionExpiration	InsufficientSessionExpiration: Checks that session expiration interval does not exceed a limit
High	Cookies In Security Decision	OPT.GO.SECURITY.CookiesInSecurityDecision	CookiesInSecurityDecision: Reliance on Cookies without Validation and Integrity Checking in a Security Decision
High	Cross Site Request Forgery	OPT.GO.SECURITY.CrossSiteRequestForgery	CrossSiteRequestForgery: Cross-site request forgery (CSRF)
High	Http Parameter Pollution	OPT.GO.SECURITY.HttpParameterPollution	HttpParameterPollution: HTTP parameter pollution (HPP)

Severity	Contrast rule	Engine rule ID	Description
High	JSON Injection	OPT.GO.SECURITY.JSONInjection	JSONInjection: Avoid using non-neutralized user-controlled input into JSON entities - JSON Injection
High	Log Forging	OPT.GO.SECURITY.LogForging	LogForging: Improper Output Neutralization for Logs
High	Open Redirect	OPT.GO.SECURITY.OpenRedirect	OpenRedirect: URL Redirection to Untrusted Site ('Open Redirect')
High	Resource Injection	OPT.GO.SECURITY.ResourceInjection	ResourceInjection: Improper control of resource identifiers ("Resource Injection")
High	Server Side Request Forgery	OPT.GO.SECURITY.ServerSideRequestForgery	ServerSideRequestForgery: Creation of requests from a vulnerable server using untrusted input (server side request forgery, SSRF)
High	Trust Boundary Violation	OPT.GO.SECURITY.TrustBoundaryViolation	TrustBoundaryViolation: Trust boundary violation
High	Unsafe Reflection	OPT.GO.SECURITY.UnsafeReflection	UnsafeReflection: Use of Externally-Controlled Input to Select Classes or Code ('Unsafe Reflection')
High	Xslt Injection	OPT.GO.SECURITY.XsltInjection	XsltInjection: XML Injection (aka Blind XPath Injection)
High	User Controlled SQL Primary Key	OPT.GO.SECURITY.UserControlledSQLPrimaryKey	UserControlledSQLPrimaryKey: Avoid using an user controlled Primary Key into a query
High	Hardcoded Ip	OPT.GO.SECURITY.HardcodedIp	HardcodedIp: Do not write IP address in source code
High	Hardcoded Salt	OPT.GO.SECURITY.HardcodedSalt	HardcodedSalt: A hardcoded salt can compromise system security
High	Insecure Transport	OPT.GO.SECURITY.InsecureTransport	InsecureTransport: Insecure transport
High	Insufficient Key Size	OPT.GO.SECURITY.InsufficientKeySize	InsufficientKeySize: Weak cryptography, insufficient key length
High	Server Insecure Transport	OPT.GO.SECURITY.ServerInsecureTransport	ServerInsecureTransport: Insecure transport in HTTP servers
Low	Password In Comments	OPT.GO.SECURITY.PasswordInComments	PasswordInComments: Storing passwords or password details in plaintext anywhere in the system or system code can compromise system security
Medium	Plaintext Storage In A Cookie	OPT.GO.SECURITY.PlaintextStorageInACookie	PlaintextStorageInACookie: Cleartext Storage of Sensitive Information in a Cookie
Medium	Unsafe Cookie	OPT.GO.SECURITY.UnsafeCookie	UnsafeCookie: Generate server-side cookies with adequate security properties
Medium	Unreachable Code	OPT.GO.RELIABILITY.UnreachableCode	UnreachableCode: Unreachable ("dead") code.
Medium	Avoid Native Calls	OPT.GO.SECURITY.AvoidNativeCalls	AvoidNativeCalls: Avoid calls from GO to C native code
Medium	Execution After Redirect	OPT.GO.SECURITY.ExecutionAfterRedirect	ExecutionAfterRedirect: Execution After Redirect (EAR)
Medium	Avoid Host Name Checks	OPT.GO.SECURITY.AvoidHostNameChecks	AvoidHostNameChecks: Avoid checks on client-side hostname, that are not reliable due to DNS poisoning
Medium	Format String Injection	OPT.GO.SECURITY.FormatStringInjection	FormatStringInjection: Exclude unsanitized user input from format strings
Medium	Potential Blocker Stmt	OPT.GO.SECURITY.PotentialBlockerStmt	PotentialBlockerStmt: Review statements that could lead to a resource exhaustion.

Severity	Contrast rule	Engine rule ID	Description
Medium	Potential Infinite Loop	OPT.GO.SECURITY.PotentialInfiniteLoop	PotentialInfiniteLoop: Loop with Unreachable Exit Condition ('Infinite Loop')
Medium	Profiling Endpoint Exposed	OPT.GO.SECURITY.ProfilingEndpointExposed	ProfilingEndpointExposed: Profiling endpoint automatically exposed
Medium	Unchecked Input In Loop Condition	OPT.GO.SECURITY.UncheckedInputInLoopCondition	UncheckedInputInLoopCondition: Unchecked input in loop condition
Medium	Hardcoded Username Password	OPT.GO.SECURITY.HardcodedUsernamePassword	HardcodedUsernamePassword: Use of Hard-coded Credentials
Medium	JSON P Hijacking	OPT.GO.SECURITY.JSONPHijacking	JSONPHijacking: Sensitive information exposed through JSONP
Medium	Password In Configuration File	OPT.GO.SECURITY.PasswordInConfigurationFile	PasswordInConfigurationFile: Use of credentials into configuration file
Medium	Plaintext Storage Of Password	OPT.GO.SECURITY.PlaintextStorageOfPassword	PlaintextStorageOfPassword: Plaintext Storage of a Password
Medium	Privacy Violation	OPT.GO.SECURITY.PrivacyViolation	PrivacyViolation: Exposure of Private Information ('Privacy Violation')
Medium	Serializable Type Containing Sensitive Data	OPT.GO.SECURITY.SerializableTypeContainingSensitiveData	SerializableTypeContainingSensitiveData: Serializable Type Containing Sensitive Data
Medium	Insecure Randomness	OPT.GO.SECURITY.InsecureRandomness	InsecureRandomness: Standard pseudo-random number generators cannot withstand cryptographic attacks

HTML Scan rules

Contrast Scan supports these rules for HTML.

Severity	Contrast rule	Engine rule ID	Description
Critical	Sandbox Allow Scripts And Same Origin	OPT.HTML.SandboxAllowScriptsAndSameOrigin	SandboxAllowScriptsAndSameOrigin: Unsafe sandbox with allow-scripts and allow-same-origin
Critical	Avoid Long Scripts In Pages	OPT.HTML.FORMATO.AvoidLongScriptsInPages	AvoidLongScriptsInPages: Avoid long js scripts
Critical	All HTML pages must be in the / docs folder	OPT.HTML.OPTIMYTH_HTML.DOCS	DOCS: ALL html pages must be in the / docs folder
Critical	Defer In Script Tag	OPT.HTML.OPTIMYTH_HTML.DeferInScriptTag	DeferInScriptTag: Use of defer attribute in script tags
Critical	Link To Js	OPT.HTML.OPTIMYTH_HTML.LinkToJs	LinkToJs: Too many references to external JavaScript files
Critical	Script Tag Position	OPT.HTML.OPTIMYTH_HTML.ScriptTagPosition	ScriptTagPosition: 'script' tag inside 'body' tag
Critical	Separate Content And Presentation	OPT.HTML.OPTIMYTH_HTML.SeparateContentAndPresentation	SeparateContentAndPresentation: Do not use JavaScript event handlers in html tags
Critical	Pages should not exceed 100Kb	OPT.HTML.OPTIMYTH_HTML.TAM	TAM: CAPTIONVALIGNb
Critical	Missing Password Field Masking	OPT.HTML.MissingPasswordFieldMasking	MissingPasswordFieldMasking: Password input field is not masked
Critical	Password In Http Get	OPT.HTML.PasswordInHttpGet	PasswordInHttpGet: Password in GET FORM

Severity	Contrast rule	Engine rule ID	Description
High	Path Relative Stylesheet Import	OPT.HTML.PathRelativeStylesheetImport	PathRelativeStylesheetImport: Path-Relative Stylesheet Import.
High	Target Blank Vulnerability	OPT.HTML.TargetBlankVulnerability	TargetBlankVulnerability: Improper Neutralization of links to external sites
Info	Form Validation Off	OPT.HTML.FormValidationOff	FormValidationOff: Form validation disabled
Info	SIZE attribute required in BASEFONT element	OPT.HTML.FORMATO.BFS	BFS: SIZE attribute required in BASEFONT element
Info	ACTION attribute required	OPT.HTML.FORMULARIOS.ACTN	ACTN: ACTION attribute required
Info	ALT attribute required	OPT.HTML.FORMULARIOS.ALT2	ALT2: ALT attribute required
Info	Wrong TYPE attribute value	OPT.HTML.FORMULARIOS.BTPE	BTPE: Wrong TYPE attribute value
Info	NAME attribute required	OPT.HTML.FORMULARIOS.NAME	NAME: NAME attribute is required
Info	TEXTAREA COLS attribute is required	OPT.HTML.FORMULARIOS.TACO	TACO: The TEXTAREA COLS attribute is missed
Info	TEXTAREA ROWS attribute is required	OPT.HTML.FORMULARIOS.TARO	TARO: TEXTAREA element without ROWS attribute
Info	VALUE attribute required	OPT.HTML.FORMULARIOS.VALU	VALU: VALUE attribute not found
Info	HEIGHT and WIDTH attributes required	OPT.HTML.GENERALES.HEWI	HEWI: HEIGHT and WIDTH attributes required
Info	BLINK element found	OPT.HTML.GENERALES.NOBLINK	NOBLINK: BLINK element found
Info	MARQUEE element found	OPT.HTML.GENERALES.NOMARQUEE	NOMARQUEE: MARQUEE element found
Info	SRC attribute not found	OPT.HTML.GENERALES.SRCC	SRCC: SRC attribute not found
Info	Incorrect TYPE attribute in OL element	OPT.HTML.LISTAS.TYPEOL	TYPEOL: Incorrect TYPE attribute in ol element
Info	Incorrect TYPE attribute in UL element	OPT.HTML.LISTAS.TYPEUL	TYPEUL: Incorrect TYPE attribute in ul element
Info	FRAMEBORDER incorrect	OPT.HTML.MARCOS.FRAMEBORDER	FRAMEBORDER: Incorrect FRAMEBORDER
Info	FRAMESET ROWS or COLS attribute missing	OPT.HTML.MARCOS.FRCR	FRCR: FRAMESET attributes missed
Info	SCROLLING attribute incorrect	OPT.HTML.MARCOS.SCROLLING	SCROLLING: SCROLLING attribute has an incorrect value
Info	No header comment found	OPT.HTML.OPTIMYTH_HTML.CBCR	CBCR: Use a header comment for every page
Info	Incorrect VALIGN attribute	OPT.HTML.TABLAS.CAPTIONVALIGN	CAPTIONVALIGN: incorrect VALIGN attribute
Info	Incorrect CLEAR attribute	OPT.HTML.TEXTO.BRCLEAR	BRCLEAR: Incorrect CLEAR attribute
Info	ALT attribute required	OPT.HTML.VARIOUS.ALT1	ALT1: ALT attribute required
Info	ALT attribute required	OPT.HTML.VARIOUS.ALT3	ALT3: ALT attribute required
Info	Area Shape	OPT.HTML.VARIOUS.AreaShape	AreaShape: incorrect SHAPE attribute

Severity	Contrast rule	Engine rule ID	Description
Info	AREA coordinates not defined	OPT.HTML.VARIOUS.CAREA	CAREA: AREA coordinates non defined
Info	PARAM VALUE attribute required E T Y P E	OPT.HTML.VARIOUS.PARAMVALUETYPE	PARAMVALUETYPE: Wrong or not specified VALUETYPE
Info	Incomplete A element	OPT.HTML.VINCULOS.AINCOMPLETO	AINCOMPLETO: Incomplete A element
Info	LINK without title attribute	OPT.HTML.VINCULOS.TLINK	TLINK: Link without 'title' attribute.
Low	Form Without Captcha	OPT.HTML.FormWithoutCaptcha	FormWithoutCaptcha: Form without CAPTCHA
Low	Add Label For Input Field	OPT.HTML.AddLabelForInputField	AddLabelForInputField: Add a label element for every input element
Low	File Upload Enabled	OPT.HTML.FileUploadEnabled	FileUploadEnabled: File upload enabled
Low	Nested Divs	OPT.HTML.NestedDivs	NestedDivs: Avoid using too many nested div elements
Low	Use descriptive comments	OPT.HTML.OPTIMYTH_HTML.CMNT	CMNT: Use descriptive comments in the pages
Low	No Javascript	OPT.HTML.OPTIMYTH_HTML.NoJavascript	NoJavascript: Javascript code within html file
Medium	Specify Integrity Attribute	OPT.HTML.SpecifyIntegrityAttribute	SpecifyIntegrityAttribute: Specify a integrity attribute on the <script> and <link> elements
Medium	Avoid Inline Styles	OPT.HTML.AvoidInlineStyles	AvoidInlineStyles: Avoid inline styles declaration
Medium	Avoid Size Attribute On Input Fields	OPT.HTML.AvoidSizeAttributeOnInputFields	AvoidSizeAttributeOnInputFields: Avoid size attribute on input fields
Medium	Embed Youtube Videos Using Iframe	OPT.HTML.EmbedYoutubeVideosUsingIframe	EmbedYoutubeVideosUsingIframe: Embed Youtube videos into an iFrame
Medium	Obsolete Attributes	OPT.HTML.ObsoleteAttributes	ObsoleteAttributes: Avoid using HTML 5 obsolete attributes
Medium	Obsolete Elements	OPT.HTML.ObsoleteElements	ObsoleteElements: Avoid using HTML 5 obsolete elements
Medium	Use external CSS files	OPT.HTML.OPTIMYTH_HTML.EUCSS	EUCSS: Limited use of CSS stylesheets in HTML pages
Medium	Noscript Tag	OPT.HTML.OPTIMYTH_HTML.NoscriptTag	NoscriptTag: Use of noscript tag
Medium	Provide Fallbacks For Multimedia Elements	OPT.HTML.ProvideFallbacksForMultimediaElements	ProvideFallbacksForMultimediaElements: Provide fallback for multimedia elements
Medium	Scripts At The Bottom	OPT.HTML.ScriptsAtTheBottom	ScriptsAtTheBottom: Put scripts at the bottom of the html body
Medium	Specify Character Encoding	OPT.HTML.SpecifyCharacterEncoding	SpecifyCharacterEncoding: Indicate the character encoding used
Medium	Specify Lang Attribute	OPT.HTML.SpecifyLangAttribute	SpecifyLangAttribute: Specify a lang attribute on the root <html> element
Medium	Stylesheets At The Top	OPT.HTML.StylesheetsAtTheTop	StylesheetsAtTheTop: Avoid importing styles in the html body
Medium	Use Doc Type	OPT.HTML.UseDocType	UseDocType: Always include a doctype declaration
Medium	Use Link For C S S Resources	OPT.HTML.UseLinkForCSSResources	UseLinkForCSSResources: Avoid using @import for CSS resources
Medium	Use S E O Relevant Meta Tags	OPT.HTML.UseSEORElevantMetaTags	UseSEORElevantMetaTags: Use relevant html meta tags for search engines

Severity	Contrast rule	Engine rule ID	Description
Medium	Should Use Content Security Policy	OPT.HTML.CORDOVA.ShouldUseContentSecurityPolicy	ShouldUseContentSecurityPolicy: Add a CSP to every page
Medium	Autocomplete On For Sensitive Fields	OPT.HTML.AutocompleteOnForSensitiveFields	AutocompleteOnForSensitiveFields: Autocomplete enabled for sensitive form fields

Informix Scan rules

Contrast Scan supports these rules for Informix.

Severity	Contrast rule	Engine rule ID	Description
Critical	Avoid Correlated Sub Selects	OPT.INFORMIX.AvoidCorrelatedSubSelects	AvoidCorrelatedSubSelects: Avoid nested SELECTs that use columns defined in outer SELECTs
Critical	Cursor For Update Where Current	OPT.INFORMIX.CursorForUpdateWhereCurrent	CursorForUpdateWhereCurrent: If a CURSOR is declared FOR UPDATE, DELETE and UPDATE must be used with the WHERE CURRENT specification
Critical	Detect Unaware Cross Joins	OPT.INFORMIX.DetectUnawareCrossJoins	DetectUnawareCrossJoins: Do not make "unnoticed" cartesian products in queries
Critical	Dont Select Known Fields	OPT.INFORMIX.DontSelectKnownFields	DontSelectKnownFields: SELECT queries never should get fields used in the WHERE specification with {}
Critical	Fetch And Declare Same Fields	OPT.INFORMIX.FetchAndDeclareSameFields	FetchAndDeclareSameFields: The number of fields to retrieve specified in the DECLARE CURSOR statement must be the same as the number of fields specified in the FETCH statement
High	Avoid Declared Unopened Cursors	OPT.INFORMIX.AvoidDeclaredUnopenedCursors	AvoidDeclaredUnopenedCursors: If a CURSOR is declared, it must be opened
High	Avoid Numeric References In By Clauses	OPT.INFORMIX.AvoidNumericReferencesInByClauses	AvoidNumericReferencesInByClauses: Do not refer to column names with number indexes in * BY clauses
High	Avoid Opened Unclosed Cursors	OPT.INFORMIX.AvoidOpenedUnclosedCursors	AvoidOpenedUnclosedCursors: If a CURSOR is opened, it must be closed
High	Avoid Opened Unused Cursors	OPT.INFORMIX.AvoidOpenedUnusedCursors	AvoidOpenedUnusedCursors: If a CURSOR is opened, it must be used
High	Avoid Union	OPT.INFORMIX.AvoidUnion	AvoidUnion: Avoid selects with UNION
High	No Current Clause	OPT.INFORMIX.NoCurrentClause	NoCurrentClause: SQL queries with CURRENT clause are heavy-weighted and must be used only when necessary
High	Unused Local Var	OPT.INFORMIX.UnusedLocalVar	UnusedLocalVar: Avoid unused local variables
High	Unused Parameter	OPT.INFORMIX.UnusedParameter	UnusedParameter: Avoid unused function or procedure parameter

Severity	Contrast rule	Engine rule ID	Description
High	Use The As Keyword	OPT.INFORMIX.UseTheAsKeyword	UseTheAsKeyword: Use AS keyword when establishing an alias to tables
Info	Avoid Concat Operator	OPT.INFORMIX.AvoidConcatOperator	AvoidConcatOperator: Do not use concatenation operator
Info	Avoid Non Declared Cursor	OPT.INFORMIX.AvoidNonDeclaredCursor	AvoidNonDeclaredCursor: Use of cursor not previously declared
Low	Avoid Goto Statement	OPT.INFORMIX.AvoidGotoStatement	AvoidGotoStatement: Avoid using GOTO statement
Low	Avoid Host Operations	OPT.INFORMIX.AvoidHostOperations	AvoidHostOperations: Do not perform arithmetic operations in the WHERE clause
Low	Avoid Insert Without Fields Specification	OPT.INFORMIX.AvoidInsertWithoutFieldsSpecification	AvoidInsertWithoutFieldsSpecification: Every INSERT statement must include the field specification (i.e : INSERT INTO table(column1,column2) VALUES (value1,value2))
Low	Avoid Scroll Cursors	OPT.INFORMIX.AvoidScrollCursors	AvoidScrollCursors: Avoid scroll cursors when possible
Low	Avoid Update Asterisk	OPT.INFORMIX.AvoidUpdateAsterisk	AvoidUpdateAsterisk: Avoid UPDATE with asterisk target
Low	Avoid Whenever	OPT.INFORMIX.AvoidWhenever	AvoidWhenever: Do not use WHENEVER clause
Low	Check Simple Condition	OPT.INFORMIX.CheckSimpleCondition	CheckSimpleCondition: Simplify WHERE clause conditions
Low	Dead Code	OPT.INFORMIX.DeadCode	DeadCode: Check and remove unreachable code
Low	Do Not Use Negation In Where	OPT.INFORMIX.DoNotUseNegationInWhere	DoNotUseNegationInWhere: Do not use negation operator
Low	Else In Case Statement	OPT.INFORMIX.ElseInCaseStatement	ElseInCaseStatement: Include an ELSE clause in CASE statements
Low	Insert Cursor In Loop	OPT.INFORMIX.InsertCursorInLoop	InsertCursorInLoop: Use INSERT cursors within loops
Low	Nested If Statements	OPT.INFORMIX.NestedIfStatements	NestedIfStatements: Avoid too deep IF statements nesting
Low	One SQL Statement Per Line	OPT.INFORMIX.OneSQLStatementPerLine	OneSQLStatementPerLine: Only write a SQL statement per line
Low	Too Many Cases In Case	OPT.INFORMIX.TooManyCasesInCase	TooManyCasesInCase: Avoid a high number of WHEN clauses in a CASE statement
Low	Too Many Lines In Function	OPT.INFORMIX.TooManyLinesInFunction	TooManyLinesInFunction: Avoid functions or procedures with too many lines
Low	Use Let Instead Of Initialize	OPT.INFORMIX.UseLetInsteadOfInitialize	UseLetInsteadOfInitialize: Use LET instead of INITIALIZE
Medium	Avoid Natural Joins	OPT.INFORMIX.AvoidNaturalJoins	AvoidNaturalJoins: NATURAL JOINs are buggy and unmaintenable
Medium	Avoid Nested Selects	OPT.INFORMIX.AvoidNestedSelects	AvoidNestedSelects: Avoid nested selects
Medium	Avoid Queries On Many Tables	OPT.INFORMIX.AvoidQueriesOnManyTables	AvoidQueriesOnManyTables: Avoid JOIN queries referencing too many tables
Medium	Avoid Select Asterisk	OPT.INFORMIX.AvoidSelectAsterisk	AvoidSelectAsterisk: Do not use SELECT *

Severity	Contrast rule	Engine rule ID	Description
Medium	Avoid Too Many Joins	OPT.INFORMIX.AvoidTooManyJoins	AvoidTooManyJoins: Avoid queries with too many JOINS
Medium	Fully Qualified Names In Columns	OPT.INFORMIX.FullyQualifiedNamesInColumns	FullyQualifiedNamesInColumns: Use qualified names when referring to column names

Java Scan rules

Contrast Scan supports these rules for Java.

Severity	Contrast rule	Engine rule ID	Description
Critical	Use Authenticated SOAP Messages	OPT.JAVA.JAX.UseAuthenticatedSOAPMessages	UseAuthenticatedSOAPMessages: Use SOAP messages authentication
Critical	Use Encrypted SOAP Messages	OPT.JAVA.JAX.UseEncryptedSOAPMessages	UseEncryptedSOAPMessages: Use encrypted SOAP messages
Critical	Use Signed SOAP Messages	OPT.JAVA.JAX.UseSignedSOAPMessages	UseSignedSOAPMessages: Use signed SOAP messages
Critical	Acegi Insecure Channel Mixing Rule	OPT.JAVA.SEC_JAVA.AcegiInsecureChannelMixingRule	AcegiInsecureChannelMixingRule: Acegi Misconfiguration - Insecure Channel Mixing
Critical	Insecure SSL	OPT.JAVA.SEC_JAVA.InsecureSSL	InsecureSSL: Insecure SSL configuration
Critical	Too Much Origins Allowed Rule	OPT.JAVA.SEC_JAVA.TooMuchOriginsAllowedRule	TooMuchOriginsAllowedRule: CORS policy (Cross-origin resource sharing) too broad
Critical	Android SQL Injection	OPT.JAVA.ANDROID.AndroidSQLInjection	AndroidSQLInjection: Avoid SQL code formed with non neutralized user input
Critical	Content Provider Uri Injection	OPT.JAVA.ANDROID.ContentProviderUriInjection	ContentProviderUriInjection: Content Provider URI Injection
Critical	Dynamically Loading Code	OPT.JAVA.ANDROID.DynamicallyLoadingCode	DynamicallyLoadingCode: Discourage dynamically loading code
Critical	Intent Manipulation	OPT.JAVA.ANDROID.IntentManipulation	IntentManipulation: Intent Manipulation
Critical	Javascript Enabled	OPT.JAVA.ANDROID.JavascriptEnabled	JavascriptEnabled: Enabling JavaScript is not recommended
Critical	Javascript Interface Annotation	OPT.JAVA.ANDROID.JavascriptInterfaceAnnotation	JavascriptInterfaceAnnotation: Potential code injection via WebView.addJavaScriptInterface()
Critical	Code Injection Rule	OPT.JAVA.SEC_JAVA.CodeInjectionRule	CodeInjectionRule: Dynamic code injection in scripting API
Critical	Code Injection With Deserialization Rule	OPT.JAVA.SEC_JAVA.CodeInjectionWithDeserializationRule	CodeInjectionWithDeserializationRule: Dynamic code injection during XML / JSON deserialization
Critical	Command Injection Rule	OPT.JAVA.SEC_JAVA.CommandInjectionRule	CommandInjectionRule: Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
Critical	Connection String Parameter Pollution	OPT.JAVA.SEC_JAVA.ConnectionStringParameterPollution	ConnectionStringParameterPollution: Connection string polluted with untrusted input

Severity	Contrast rule	Engine rule ID	Description
Critical	Never Use Identifier In Equals Hashcode	OPT.HIBERNATE.NeverUseIdentifierInEqualsHashCode	NeverUseIdentifierInEqualsHashCode: Never ever use the database identifier in equals() and hashCode()
Critical	Rollback Transaction On Exception	OPT.HIBERNATE.RollbackTransactionOnException	RollbackTransactionOnException: Rollback transactions if an exception occurs
Critical	Static Inner Persistent Classes	OPT.HIBERNATE.StaticInnerPersistentClasses	StaticInnerPersistentClasses: If an inner class is persistent, it should be static
Critical	Access to persistence layer from Struts Actions	OPT.JAVA.ACTIONS.ALPA	ALPA: Access to persistence layer from Struts Actions
Critical	N A E A	OPT.JAVA.ACTIONS.NAEA	NAEA: Do not use non final instance fields in Actions
Critical	Activity Start At Broadcast Intent	OPT.JAVA.ANDROID.ActivityStartAtBroadcastIntent	ActivityStartAtBroadcastIntent: Activity Start in response to broadcast Intent
Critical	Cross Site Scripting Rule	OPT.JAVA.SEC_JAVA.CrossSiteScriptingRule	CrossSiteScriptingRule: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
Critical	Http Splitting Rule	OPT.JAVA.SEC_JAVA.HttpSplittingRule	HttpSplittingRule: Improper Neutralization of CRLF Sequences in HTTP Headers ('HTTP Response Splitting')
Critical	I Batis SQL Injection Rule	OPT.JAVA.SEC_JAVA.iBatisSqlInjectionRule	iBatisSqlInjectionRule: Improper Neutralization of Special Elements used in an SQL Command in iBatis ('SQL Injection')
Critical	Ldap Injection Rule	OPT.JAVA.SEC_JAVA.LdapInjectionRule	LdapInjectionRule: Avoid non-neutralized user-controlled input in LDAP search filters
Critical	Mail Command Injection	OPT.JAVA.SEC_JAVA.MailCommandInjection	MailCommandInjection: Mail Command Injection
Critical	No SQL Injection	OPT.JAVA.SEC_JAVA.NoSQLInjection	NoSQLInjection: Improper neutralization of special elements in data query logic (NoSQL injection)
Critical	Process Control Rule	OPT.JAVA.SEC_JAVA.ProcessControlRule	ProcessControlRule: Library loaded from untrusted source
Critical	Regex Injection Rule	OPT.JAVA.SEC_JAVA.RegexInjectionRule	RegexInjectionRule: Prevent denial of service attack through malicious regular expression ('Regex Injection')
Critical	Privilege Escalation Attack	OPT.JAVA.ANDROID.PrivilegeEscalationAttack	PrivilegeEscalationAttack: Don't allow applications to execute code using other applications privileges
Critical	Same Origin Method Execution	OPT.JAVA.SEC_JAVA.SameOriginMethodExecution	SameOriginMethodExecution: Same Origin Method Execution (SOME)
Critical	Server Side Request Forgery Rule	OPT.JAVA.SEC_JAVA.ServerSideRequestForgeryRule	ServerSideRequestForgeryRule: Server-Side Request Forgery (SSRF)
Critical	Spring View Manipulation	OPT.JAVA.SEC_JAVA.SpringViewManipulation	SpringViewManipulation: Spring View Manipulation
Critical	S Q L Resources	OPT.JAVA.ANDROID.SQLResources	SQLResources: Close SQL resources when finishing using
Critical	SQL Injection Rule	OPT.JAVA.SEC_JAVA.SqlInjectionRule	SqlInjectionRule: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
Critical	B L N C	OPT.JAVA.BEANS.BLNC	BLNC: Use appropriate signatures for all listener methods in JavaBeans
Critical	Avoid Assign In For	OPT.JAVA.BUC.AvoidAssignInFor	AvoidAssignInFor: Avoid assigning values to variables inside for loops
Critical	Avoid Reuse Var	OPT.JAVA.BUC.AvoidReuseVar	AvoidReuseVar: Avoid reusing the control variables in nested loops

Severity	Contrast rule	Engine rule ID	Description
Critical	Float Double Comparison	OPT.JAVA.COMP.FloatDoubleComparison	FloatDoubleComparison: Avoid using Code Quality[{'CERT-J': ['NUM07-J']}]
Critical	Avoid Call AWT From Servlet	OPT.JAVA.COMPDAT.AvoidCallAWTFromServlet	AvoidCallAWTFromServlet: Avoid calling Java.awt.* from servlet code
Critical	Avoid Call Swing From Servlet	OPT.JAVA.COMPDAT.AvoidCallSwingFromServlet	AvoidCallSwingFromServlet: Avoid calling Javax.swing.* from any servlet
Critical	Avoid Return In Constructors	OPT.JAVA.CONST.AvoidReturnInConstructors	AvoidReturnInConstructors: Avoid constructors with return types
Critical	Avoid Declare Matrix Volatile	OPT.JAVA.DECL.AvoidDeclareMatrixVolatile	AvoidDeclareMatrixVolatile: Do not declare an array field as volatile
Critical	Avoid Inaccessible Class	OPT.JAVA.DECL.AvoidInaccessibleClass	AvoidInaccessibleClass: Avoid inaccessible classes
Critical	Avoid Break Continue In Labels	OPT.JAVA.ESTRUC.AvoidBreakContinueInLabels	AvoidBreakContinueInLabels: Avoid using continue and break with labels
Critical	Avoid Call Sound From Servlet	OPT.JAVA.EXAC.AvoidCallSoundFromServlet	AvoidCallSoundFromServlet: Avoid calling Javax.sound.* from any servlet code
Critical	Avoid Throw Null Pointer Exceptions	OPT.JAVA.EXCP.AvoidThrowNullPointerExceptions	AvoidThrowNullPointerExceptions: Avoid launching NullPointerExceptions
Critical	Avoid Remove Action Listener	OPT.JAVA.FIN.AvoidRemoveActionListener	AvoidRemoveActionListener: Avoid removing listeners in finalize() methods
Critical	A B C L	OPT.JAVA.FMETODOS.ABCL	ABCL: Avoid usage of break/continue sentences with a label
Critical	A GC	OPT.JAVA.GC.AGC	AGC: Do not call to GC with System.gc()
Critical	I F F	OPT.JAVA.GC.IFF	IFF: Call super.finalize() in the finally block of finalize() methods
Critical	Avoid Await Outside Brackets	OPT.JAVA.HEB.AvoidAwaitOutsideBrackets	AvoidAwaitOutsideBrackets: Avoid using the method await() of the class java.util.concurrent.locks.Condition class outside a while loop
Critical	Avoid Call Wait	OPT.JAVA.HEB.AvoidCallWait	AvoidCallWait: Avoid calling wait() on an object java.util.concurrent.locks.Condition
Critical	Avoid Runnable Without Run	OPT.JAVA.HEB.AvoidRunnableWithoutRun	AvoidRunnableWithoutRun: Avoid implementing Runnable without a run() method
Critical	Run With Synchronize	OPT.JAVA.HEB.RunWithSynchronize	RunWithSynchronize: Always use 'synchronized' with the run() methods
Critical	Avoid Load Library	OPT.JAVA.J2EE.AvoidLoadLibrary	AvoidLoadLibrary: Avoid calling java.lang.System.loadLibrary() or java.lang.Runtime.loadLibrary()
Critical	Avoid Set Security Manager	OPT.JAVA.J2EE.AvoidSetSecurityManager	AvoidSetSecurityManager: Avoid calling java.lang.System.setSecurityManager()
Critical	Avoid Protected Native Methods	OPT.JAVA.J2SE.AvoidProtectedNativeMethods	AvoidProtectedNativeMethods: Avoid protected native methods
Critical	Avoid Public Native Methods	OPT.JAVA.J2SE.AvoidPublicNativeMethods	AvoidPublicNativeMethods: Avoid public native methods
Critical	Avoid Invalid Exception Handling	OPT.JAVA.JAX.AvoidInvalidExceptionHandling	AvoidInvalidExceptionHandling: Perform a exception handling conforming to JAX-WS specifications

Severity	Contrast rule	Engine rule ID	Description
Critical	Avoid Web Method Annotation In Endpoint Interfaces	OPT.JAVA.JAX.AvoidWebMethodAnnotationInEndpointInterfaces	AvoidWebMethodAnnotationInEndpointInterfaces: Avoid using @WebMethod annotation in endpoint interfaces methods
Critical	Xml Entity Injection Rule	OPT.JAVA.SEC_JAVA.XmlEntityInjectionRule	XmlEntityInjectionRule: XML entity injection
Critical	Avoid Data Submission To Non Editable Field	OPT.JAVA.SPRING.AvoidDataSubmissionToNonEditableField	AvoidDataSubmissionToNonEditableField: Avoid data submissions to non editable fields
Critical	D S L V	OPT.JAVA.JDBC.DSLV	DSLVS: Do not create Datasource variables in local methods
Critical	S E T P S	OPT.JAVA.JDBC.SETPS	SETPS: PreparedStatement parameter values must be defined before executing the query
Critical	Set Up J Unit	OPT.JAVA.JUNIT_JAVA.SetUpJUnit	SetUpJUnit: Always overwrite the method setUp() in each test case of JUnits
Critical	Avoid Call Thread From Servlet	OPT.JAVA.MAN.AvoidCallThreadFromServlet	AvoidCallThreadFromServlet: Avoid calling java.lang.Thread from any servlet
Critical	Avoid Call Finalize	OPT.JAVA.MEM.AvoidCallFinalize	AvoidCallFinalize: Avoid calling finalize() except in the finally block of the method finalize()
Critical	Avoid Garbage Collector	OPT.JAVA.MEM.AvoidGarbageCollector	AvoidGarbageCollector: Avoid invoking the garbage collector
Critical	Avoid Publicfinalize	OPT.JAVA.MEM.AvoidPublicfinalize	AvoidPublicfinalize: Do not declare finalize() public
Critical	A S I	OPT.JAVA.PB.ASI	ASI: Possible confusion between assignment and comparison in a conditional expression
Critical	M A I L	OPT.JAVA.PB.MAIL	MAIL: Avoid usages of javax.mail.*
Critical	N A M I N G	OPT.JAVA.PB.NAMING	NAMING: Avoid naming not-constructor methods with the same name than its class
Critical	T L S	OPT.JAVA.PB.TLS	TLS: Avoid using text tags in statements
Critical	Avoid Call Applet From Servlet	OPT.JAVA.RECBAS.AvoidCallAppletFromServlet	AvoidCallAppletFromServlet: Avoid calling java.applet.* from the finalize() method of a servlet
Critical	Avoid Call Class Loader From Servlet	OPT.JAVA.RECBAS.AvoidCallClassLoaderFromServlet	AvoidCallClassLoaderFromServlet: Avoid calling java.lang.ClassLoader from finalize() method of a servlet
Critical	Avoid Call Context From Servlet	OPT.JAVA.RECBAS.AvoidCallContextFromServlet	AvoidCallContextFromServlet: Avoid calling javax.naming.Context from the finalize() method of a servlet
Critical	Avoid Call Driver From Servlet	OPT.JAVA.RECBAS.AvoidCallDriverFromServlet	AvoidCallDriverFromServlet: Avoid calling java.sql.Driver from the finalize() method of a servlet
Critical	Avoid Call I O From Servlet	OPT.JAVA.RECBAS.AvoidCallIOFromServlet	AvoidCallIOFromServlet: Avoid calling java.io.* from the finalize() method of a servlet
Critical	Avoid Call Lang Ref From Servlet	OPT.JAVA.RECBAS.AvoidCallLangRefFromServlet	AvoidCallLangRefFromServlet: Avoid calling java.lang.ref.* from finalize() method of a servlet
Critical	Avoid Call Port Rem Obj From Servlet	OPT.JAVA.RECBAS.AvoidCallPortRemObjFromServlet	AvoidCallPortRemObjFromServlet: Avoid calling javax.rmi.PortableRemoteObject from the finalize() method of a servlet
Critical	Avoid Call Runnable From Servlet	OPT.JAVA.RECBAS.AvoidCallRunnableFromServlet	AvoidCallRunnableFromServlet: Avoid calling java.lang.Runnable from the finalize() method of a servlet
Critical	Avoid Call Security From Servlet	OPT.JAVA.RECBAS.AvoidCallSecurityFromServlet	AvoidCallSecurityFromServlet: Avoid calling java.security.* from finalize() method of a servlet
Critical	Avoid Call Transaction From Servlet	OPT.JAVA.RECBAS.AvoidCallTransactionFromServlet	AvoidCallTransactionFromServlet: Avoid calling javax.transaction.* from the finalize() method of a servlet

Severity	Contrast rule	Engine rule ID	Description
Critical	Avoid Get Declared Method	OPT.JAVA.REFL.AvoidGetDeclaredMethod	AvoidGetDeclaredMethod: Avoid calling java.lang.Class.getDeclaredMethod()
Critical	Avoid Get Field	OPT.JAVA.REFL.AvoidGetField	AvoidGetField: Avoid calling java.lang.Class.getField()
Critical	Avoid Get Method	OPT.JAVA.REFL.AvoidGetMethod	AvoidGetMethod: Avoid calling the method getMethod() from java.lang.Class
Critical	Avoid Call Compiler From Servlet	OPT.JAVA.RENDESC.AvoidCallCompilerFromServlet	AvoidCallCompilerFromServlet: Avoid calling java.lang.Compiler from any servlet
Critical	Avoid Call Reflect From Servlet	OPT.JAVA.RENDESC.AvoidCallReflectFromServlet	AvoidCallReflectFromServlet: Avoid calling Java.lang.reflect.* from any servlet
Critical	Avoid Call Runtime From Servlet	OPT.JAVA.RENDESC.AvoidCallRuntimeFromServlet	AvoidCallRuntimeFromServlet: Avoid calling java.lang.Runtime from any servlet
Critical	Avoid Call Thread Group From Servlet	OPT.JAVA.RENDESC.AvoidCallThreadGroupFromServlet	AvoidCallThreadGroupFromServlet: Avoid calling java.lang.ThreadGroup from any servlet
Critical	Avoid Call Zip From Servlet	OPT.JAVA.RENDESC.AvoidCallZipFromServlet	AvoidCallZipFromServlet: Avoid calling Java.util.zip.* from any servlet
Critical	E A O F	OPT.JAVA.RGME.EAOF	EAOF: Do not access a class field too many times
Critical	Dont Use Keywords	OPT.JAVA.RGP.DontUseKeywords	DontUseKeywords: Do not use keywords in later versions of the language
Critical	E N V	OPT.JAVA.RGP.ENV	ENV: Avoid using System.getenv()
Critical	E R A	OPT.JAVA.RGP.ERA	ERA: Avoid absolute paths
Critical	E X E C	OPT.JAVA.RGP.EXEC	EXEC: Avoid using Runtime.exec()
Critical	N A T V	OPT.JAVA.RGP.NATV	NATV: Avoid user-defined native methods
Critical	P E E R	OPT.JAVA.RGP.PEER	PEER: Avoid using java.awt.peer.* interfaces
Critical	Accessibility Subversion Rule	OPT.JAVA.SEC_JAVA.AccessibilitySubversionRule	AccessibilitySubversionRule: Java access restriction subverted (Reflection)
Critical	Acegi Run As Authentication Replacement Rule	OPT.JAVA.SEC_JAVA.AcegiRunAsAuthenticationReplacementRule	AcegiRunAsAuthenticationReplacementRule: Acegi Misconfiguration - Run-As Authentication Replacement
Critical	Anonymous Ldap Bind Rule	OPT.JAVA.SEC_JAVA.AnonymousLdapBindRule	AnonymousLdapBindRule: Access Control - Anonymous LDAP Bind
Critical	Path Traversal Rule	OPT.JAVA.SEC_JAVA.PathTraversalRule	PathTraversalRule: Avoid non-neutralized user-controlled input composed in a pathname to a resource
Critical	Spring Unrestricted Request Mapping	OPT.JAVA.SEC_JAVA.SpringUnrestrictedRequestMapping	SpringUnrestrictedRequestMapping: Spring CSRF unrestricted RequestMapping.
Critical	Static Database Connection	OPT.JAVA.SEC_JAVA.StaticDatabaseConnection	StaticDatabaseConnection: Static database connection / session
Critical	Avoid Add Container Himself	OPT.JAVA.SENT.AvoidAddContainerHimself	AvoidAddContainerHimself: Avoid adding a container itself
Critical	Avoid Incorrect Increase	OPT.JAVA.SENT.AvoidIncorrectIncrease	AvoidIncorrectIncrease: Avoid assigning a post increment to itself
Critical	Avoid Invoke Exit	OPT.JAVA.SENT.AvoidInvokeExit	AvoidInvokeExit: Do not call System.exit()
Critical	Avoid Invoke Run Finalizers On Exit	OPT.JAVA.SENT.AvoidInvokeRunFinalizersOnExit	AvoidInvokeRunFinalizersOnExit: Avoid calling java.lang.System.runFinalizersOnExit()

Severity	Contrast rule	Engine rule ID	Description
Critical	Not Use Label Sentences	OPT.JAVA.SENT.NotUseLabelSentences	NotUseLabelSentences: Do not use labels
Critical	Declare Constructor For Externalizable	OPT.JAVA.SERI.DeclareConstructorForExternalizable	DeclareConstructorForExternalizable: Always declare a constructor for a class that implements java.io.Externalizable
Critical	Read Resolve Return Object	OPT.JAVA.SERI.ReadResolveReturnObject	ReadResolveReturnObject: Always have a return type of java.lang.Object() in the methods readResolve()
Critical	S B L	OPT.JAVA.STR.SBL	SBL: Remove toString method when using StringBuffer/StringBuilder to obtain string size
Critical	Avoid Sun Star	OPT.JAVA.SUN.AvoidSunStar	AvoidSunStar: Avoid using sun.*
Critical	Avoid Label Switch Sentences	OPT.JAVA.SWITCH.AvoidLabelSwitchSentences	AvoidLabelSwitchSentences: Avoid using text labels in switch commands
Critical	A U T Y	OPT.JAVA.TRS.AUTY	AUTY: Avoid using Thread.yield
Critical	C S F S	OPT.JAVA.TRS.CSFS	CSFS: Do not cause deadlocks by calling synchronized method from another synchronized method
Critical	N S Y N	OPT.JAVA.TRS.NSYN	NSYN: Avoid calling wait, notify or notifyAll out of a synchronized context
Critical	T H R D	OPT.JAVA.TRS.THRD	THRD: Avoid calling Thread.resume(), Thread.stop(), Thread.suspend(), or Runtime.runFinalizersOnExit()
Critical	Android Sticky Broadcast	OPT.JAVA.ANDROID.AndroidStickyBroadcast	AndroidStickyBroadcast: Avoid Sticky Broadcast
Critical	Receiver Without Permission	OPT.JAVA.ANDROID.ReceiverWithoutPermission	ReceiverWithoutPermission: Missing broadcast permission when register a receiver
Critical	SMS Monitoring	OPT.JAVA.ANDROID.SMSMonitoring	SMSMonitoring: Don't use SMS for data input or command
Critical	Password In Redirect Rule	OPT.JAVA.SEC_JAVA.PasswordInRedirectRule	PasswordInRedirectRule: Password Management - Password in Redirect
Critical	Use A Safe Cipher	OPT.JAVA.ANDROID.UseASafeCipher	UseASafeCipher: Avoid using cipher in ECB mode, or without specifying the mode
Critical	Hardcoded Crypto Key	OPT.JAVA.SEC_JAVA.HardcodedCryptoKey	HardcodedCryptoKey: Hardcoded cryptographic keys
Critical	Non Random IV With CBC Mode	OPT.JAVA.SEC_JAVA.NonRandomIVWithCBCMode	NonRandomIVWithCBCMode: Not using a Random IV with CBC Mode
Critical	Weak Cryptographic Hash Rule	OPT.JAVA.SEC_JAVA.WeakCryptographicHashRule	WeakCryptographicHashRule: Weak cryptographic hash
Critical	Weak Encryption Rule	OPT.JAVA.SEC_JAVA.WeakEncryptionRule	WeakEncryptionRule: Weak symmetric encryption algorithm
High	Do Not Release Debuggable Apps	OPT.JAVA.ANDROID.DoNotReleaseDebuggableApps	DoNotReleaseDebuggableApps: Do not release debuggable apps
High	Prevent Backup Vulnerability	OPT.JAVA.ANDROID.PreventBackupVulnerability	PreventBackupVulnerability: Inadequate backup configuration
High	Dynamic Method Invocation	OPT.JAVA.SEC_JAVA.DynamicMethodInvocation	DynamicMethodInvocation: Dynamic method invocation in Struts 2
High	Insufficient Session Expiration Rule	OPT.JAVA.SEC_JAVA.InsufficientSessionExpirationRule	InsufficientSessionExpirationRule: Checks that session expiration interval is positive and does not exceed a limit

Severity	Contrast rule	Engine rule ID	Description
High	Play Security Misconfiguration	OPT.JAVA.SEC_JAVA.PlaySecurityMisconfiguration	PlaySecurityMisconfiguration: Security misconfiguration in Play framework.
High	Web Xml Security Misconfigurations Rule	OPT.JAVA.SEC_JAVA.WebXmlSecurityMisconfigurationsRule	WebXmlSecurityMisconfigurationsRule: Avoid misconfiguring security properties in web.xml descriptor
High	Prevent Exposition Of All Repositories	OPT.JAVA.SPRING.PreventExpositionOfAllRepositories	PreventExpositionOfAllRepositories: Avoid exposing all repositories as REST resources
High	Bind Parameters In Queries	OPT.HIBERNATE.BindParametersInQueries	BindParametersInQueries: Use bind (or named) parameters in HQL and native SQL queries
High	Mock Location	OPT.JAVA.ANDROID.MockLocation	MockLocation: Avoid using a mock location provider
High	Package Manager Get Signatures	OPT.JAVA.ANDROID.PackageManagerGetSignatures	PackageManagerGetSignatures: Potential Multiple Certificate Exploit
High	Avoid Multiple Entities Mapped To Same Table	OPT.HIBERNATE.AvoidMultipleEntitiesMappedToSameTable	AvoidMultipleEntitiesMappedToSameTable: Avoid more than one entity mapped to same database table
High	Classes Should Be Their Own Proxy	OPT.HIBERNATE.ClassesShouldBeTheirOwnProxy	ClassesShouldBeTheirOwnProxy: Every persistent class should be its own proxy
High	Close Sessions Where Opened	OPT.HIBERNATE.CloseSessionsWhereOpened	CloseSessionsWhereOpened: Close sessions in the method where they are opened
High	Declare Private Identifier Setter	OPT.HIBERNATE.DeclarePrivateIdentifierSetter	DeclarePrivateIdentifierSetter: The setter method for an identifier property (id or composite-id) should be private
High	Declare Type For Date Property	OPT.HIBERNATE.DeclareTypeForDateProperty	DeclareTypeForDateProperty: Declare type for java.util.Date property in configuration file
High	Implement Zero Argument Constructor	OPT.HIBERNATE.ImplementZeroArgumentConstructor	ImplementZeroArgumentConstructor: Implement a zero-argument constructor for persistent classes
High	Invalid Property Type Mapping	OPT.HIBERNATE.InvalidPropertyTypeMapping	InvalidPropertyTypeMapping: Map a Hibernate property type only to the corresponding Java type
High	Referenced Class Not Defined	OPT.HIBERNATE.ReferencedClassNotDefined	ReferencedClassNotDefined: Classes referenced in the configuration file (*.hbm.xml) should be declared
High	Aapt Crash	OPT.JAVA.ANDROID.AaptCrash	AaptCrash: Avoid defining styles with dynamically generated identifiers
High	Adapter View Children	OPT.JAVA.ANDROID.AdapterViewChildren	AdapterViewChildren: AdapterViews cannot have children in xml
High	Always Canonicalize URL Received By Content Provider	OPT.JAVA.ANDROID.AlwaysCanonicalizeURLReceivedByContentProvider	AlwaysCanonicalizeURLReceivedByContentProvider: Avoid improper access to application data
High	Cross Site Request Forgery Rule	OPT.JAVA.SEC_JAVA.CrossSiteRequestForgeryRule	CrossSiteRequestForgeryRule: Cross-site request forgery (CSRF)
High	Call Super First On Init	OPT.JAVA.ANDROID.CallSuperFirstOnInit	CallSuperFirstOnInit: Be sure super method is called first on initialization methods

Severity	Contrast rule	Engine rule ID	Description
High	Call Super Last On End	OPT.JAVA.ANDROID.CallSuperLastOnEnd	CallSuperLastOnEnd: Ensure that on finalization methods, super method is called at the end of the method
High	External Control Of Configuration Setting	OPT.JAVA.SEC_JAVA.ExternalControlOfConfigurationSetting	ExternalControlOfConfigurationSetting: External Control of System or Configuration Setting
High	Device Admin	OPT.JAVA.ANDROID.DeviceAdmin	DeviceAdmin: Check if receiver acts like a device admin
High	Http Parameter Pollution Rule	OPT.JAVA.SEC_JAVA.HttpParameterPollutionRule	HttpParameterPollutionRule: HTTP parameter pollution (HPP)
High	Dont Use Config	OPT.JAVA.ANDROID.DontUseConfig	DontUseConfig: Avoid using deprecated android.util.Config
High	Dont Use Find View By Id Repeatedly	OPT.JAVA.ANDROID.DontUseFindViewByIdRepeatedly	DontUseFindViewByIdRepeatedly: Avoid repeatedly calling to findViewById() with the same identifier
High	Duplicate Ids	OPT.JAVA.ANDROID.DuplicateIds	DuplicateIds: Avoid duplicated ids within the same layout
High	JSON Injection	OPT.JAVA.SEC_JAVA.JSONInjection	JSONInjection: Avoid using non-neutralized user controlled input into JSON entities - JSON Injection
High	Log Forging	OPT.JAVA.SEC_JAVA.LogForging	LogForging: Improper Output Neutralization for Logs
High	Grant All Uris	OPT.JAVA.ANDROID.GrantAllUris	GrantAllUris: Avoid sharing root path
High	Incorrect Wake Lock Usage	OPT.JAVA.ANDROID.IncorrectWakeLockUsage	IncorrectWakeLockUsage: Make sure WakeLock is liberated
High	Illegal Resource Ref	OPT.JAVA.ANDROID.IllegalResourceRef	IllegalResourceRef: Checks versionCode and versionName are literals
High	Open Redirect Rule	OPT.JAVA.SEC_JAVA.OpenRedirectRule	OpenRedirectRule: URL Redirection to Untrusted Site ('Open Redirect')
High	Missing Super Call	OPT.JAVA.ANDROID.MissingSuperCall	MissingSuperCall: Check super method is called into the implementation
High	Reflected File Download	OPT.JAVA.SEC_JAVA.ReflectedFileDownload	ReflectedFileDownload: Improper Neutralization of Input leads to Reflected File Download
High	Nested Scrolling	OPT.JAVA.ANDROID.NestedScrolling	NestedScrolling: Avoid scrolling widgets with scrolling widgets children
High	Resource Injection	OPT.JAVA.SEC_JAVA.ResourceInjection	ResourceInjection: Improper control of resource identifiers ("Resource Injection")
High	Reference Type	OPT.JAVA.ANDROID.ReferenceType	ReferenceType: Alias and resource types have to be the same
High	Resource As Color	OPT.JAVA.ANDROID.ResourceAsColor	ResourceAsColor: Avoid method calls using color id directly as parameter
High	Resource Cycle	OPT.JAVA.ANDROID.ResourceCycle	ResourceCycle: Cycles can't exist in resources definition
High	Scroll View Size	OPT.JAVA.ANDROID.ScrollViewSize	ScrollViewSize: Check layout_width and layout_height attributes for ScrollView children
High	Sd Card Path	OPT.JAVA.ANDROID.SdCardPath	SdCardPath: Avoid hardcoded references to SD card path
High	Scroll View Count	OPT.JAVA.ANDROID.ScrollViewCount	ScrollViewCount: Check ScrollViews just have one children
High	Text View Edits	OPT.JAVA.ANDROID.TextViewEdits	TextViewEdits: Check if TextView is correctly used
High	Use Check Permission	OPT.JAVA.ANDROID.UseCheckPermission	UseCheckPermission: Use the result of a permission check
High	Use Serialization Judiciously	OPT.JAVA.ANDROID.UseSerializationJudiciously	UseSerializationJudiciously: Discourage use of Serialization, use JSON instead
High	Uses Min Sdk Attributes	OPT.JAVA.ANDROID.UsesMinSdkAttributes	UsesMinSdkAttributes: Check required API level are specified

Severity	Contrast rule	Engine rule ID	Description
High	Wrong View Cast	OPT.JAVA.ANDROID.WrongViewCast	WrongViewCast: Type check for views with an assigned id
High	Wait Sleep In Activit	OPT.JAVA.ANDROID.WaitSleepInActivit	WaitSleepInActivit: Avoid using Thread.wait or Thread.sleep in Activity
High	J D B C	OPT.JAVA.BEANS.JDBC	JDBC: Avoid JDBC use in Bean classes
High	S Z B L	OPT.JAVA.BEANS.SZBL	SZBL: Verify that beans implements java.io.Serializable
High	Avoid Use Replace Methods Rule	OPT.JAVA.CADCAR.AvoidUseReplaceMethodsRule	AvoidUseReplaceMethodsRule: Do not use '.' in String methods expecting a regular expression
High	P J D C C	OPT.JAVA.CDCI.PJDCC	PJDCC: Provide Javadoc comments for public classes and interfaces
High	S Y N	OPT.JAVA.CFFSERVLET.SYN	SYN: Minimize synchronization in Servlets
High	C T N L	OPT.JAVA.CMETRICS.CTNL	CTNL: Avoid classes / interfaces with too many lines of code
High	N O F	OPT.JAVA.CMETRICS.NOF	NOF: Maximum allowed number of fields
High	N O M	OPT.JAVA.CMETRICS.NOM	NOM: Maximum allowed number of methods
High	T C C	OPT.JAVA.CMETRICS.TCC	TCC: Cyclomatic complexity
High	A A I	OPT.JAVA.CNU.AAI	AAI: Avoid unnecessary modifiers in an interface
High	D I	OPT.JAVA.CNU.DI	DI: Avoid duplicated imports
High	E I	OPT.JAVA.CNU.EI	EI: Avoid excessive import lines
High	E P N U	OPT.JAVA.CNU.EPNU	EPNU: Avoid unused parameters
High	E V N U	OPT.JAVA.CNU.EVNU	EVNU: Avoid unused local variables
High	P F	OPT.JAVA.CNU.PF	PF: Avoid unused private fields
High	P M	OPT.JAVA.CNU.PM	PM: Avoid unused private methods and constructors
High	U I	OPT.JAVA.CNU.UI	UI: Avoid unused imports
High	E I S	OPT.JAVA.COL.EIS	EIS: Avoid concurrently iterations over a collection
High	Equals Hash Code	OPT.JAVA.COMP.EqualsHashCode	EqualsHashCode: Always overwrite java.lang.Object.equals() and java.lang.Object.hashCode()
High	Method Equals	OPT.JAVA.COMP.MethodEquals	MethodEquals: Make sure the method name is equals() and not equal
High	Avoid Return Object	OPT.JAVA.CONV.AvoidReturnObject	AvoidReturnObject: Avoid returning Java.lang.Object, instead convert it to a specific type
High	Avoid Method Invok Only Super Method	OPT.JAVA.DECL.AvoidMethodInvokOnlySuperMethod	AvoidMethodInvokOnlySuperMethod: Avoid methods that invoke only overwritten supermethods
High	Avoid Not Use Field	OPT.JAVA.DECL.AvoidNotUseField	AvoidNotUseField: Avoid unused fields
High	Correct Hash Code	OPT.JAVA.DECL.CorrectHashCode	CorrectHashCode: Make sure the hashCode() method is spelled correctly
High	Correct To String	OPT.JAVA.DECL.CorrectToString	CorrectToString: Make sure the name of the method is toString() and not toStringing()
High	Declare Equals Method Of Compareable	OPT.JAVA.DECL.DeclareEqualsMethodOfCompareable	DeclareEqualsMethodOfCompareable: Always declare equals() method in a class that implements java.lang.Comparable
High	Signature Standard Equals	OPT.JAVA.DECL.SignatureStandardEquals	SignatureStandardEquals: Always use the standard signature for equals methods
High	Make your clone() method final for security	OPT.JAVA.DECLARA.CLONE	CLONE: Declare clone() throws CloneNotSupportedException for Cloneable clas

Severity	Contrast rule	Engine rule ID	Description
High	Trust Boundary Violation Rule	OPT.JAVA.SEC_JAVA.TrustBoundaryViolationRule	TrustBoundaryViolationRule: Trust boundary violation
High	U C C	OPT.JAVA.DECLARA.UCC	UCC: Private constructors in class with static members
High	Avoid Remote Exception	OPT.JAVA.EJB.AvoidRemoteException	AvoidRemoteException: Throw RemoteException from methods of remote interfaces
High	Dont Avoid Remote Exception	OPT.JAVA.EJB.DontAvoidRemoteException	DontAvoidRemoteException: Do not throw an 'java.rmi.RemoteException' for methods of local interfaces
High	Public Constructor Without Parameters	OPT.JAVA.EJB.PublicConstructorWithoutParameters	PublicConstructorWithoutParameters: Every EJB class should have a public constructor with no parameters
High	Reuse EJB Home Instances	OPT.JAVA.EJB.ReuseEJBHomeInstances	ReuseEJBHomeInstances: Always reuse EJBHome instances
High	Avoid New Throwable	OPT.JAVA.EXCP.AvoidNewThrowable	AvoidNewThrowable: Avoid creating new instances of java.lang.Throwable
High	Avoid Null Pointer Exception	OPT.JAVA.EXCP.AvoidNullPointerException	AvoidNullPointerException: Avoid capturing NullPointerExceptions
High	Avoid Throw Error	OPT.JAVA.EXCP.AvoidThrowError	AvoidThrowError: Avoid throwing java.lang.Error
High	Avoid Throw Runtime Excetions	OPT.JAVA.EXCP.AvoidThrowRuntimeExcetions	AvoidThrowRuntimeExcetions: Avoid throwing RuntimeExceptions
High	Avoid Empty Methods Finalize	OPT.JAVA.FIN.AvoidEmptyMethodsFinalize	AvoidEmptyMethodsFinalize: Avoid empty finalize() methods
High	Avoid Overload Finalize	OPT.JAVA.FIN.AvoidOverloadFinalize	AvoidOverloadFinalize: Avoid overloading the method finalize()
High	Dont Call Finalize	OPT.JAVA.FIN.DontCallFinalize	DontCallFinalize: Never call finalize() explicitly
High	N C A C	OPT.JAVA.FMETODOS.NCAC	NCAC: Do not call an abstract method from a constructor in abstract classes
High	O V E R R I D E	OPT.JAVA.FMETODOS.OVERRIDE	OVERRIDE: Override Object.equals ()when you override Object.hashCode ()
High	A U T P	OPT.JAVA.GC.AUTP	AUTP: Avoid unnecessary temporal wrapping objects when converting primitive data to String
High	D U D	OPT.JAVA.GC.DUD	DUD: Avoid using Date[], use long[] instead
High	F C F	OPT.JAVA.GC.FCF	FCF: Call super.finalize() from finalize()
High	O S T M	OPT.JAVA.GC.OSTM	OSTM: Prevent potential memory leaks in ObjectOutputStreams by calling reset () or close ()
High	S T V	OPT.JAVA.GC.STV	STV: Avoid static collections; they can grow without bounds
High	Avoid Call Interrupted Object	OPT.JAVA.HEB.AvoidCallInterruptedObject	AvoidCallInterruptedObject: Avoid calling Thread.interrupted() on an arbitrary Thread
High	Avoid Call Run	OPT.JAVA.HEB.AvoidCallRun	AvoidCallRun: Avoid calling Thread.run()
High	Avoid Sleep Inside While	OPT.JAVA.HEB.AvoidSleepInsideWhile	AvoidSleepInsideWhile: Avoid using while() and sleep(), instead use wait() and notify()
High	Avoid Synchronized Blocks Notify	OPT.JAVA.HEB.AvoidSynchronizedBlocksNotify	AvoidSynchronizedBlocksNotify: Avoid synchronised blocks that have an invocation to notify() or notifyAll() as the statement
High	Avoid Synchronized Object Lock	OPT.JAVA.HEB.AvoidSynchronizedObjectLock	AvoidSynchronizedObjectLock: Avoid the synchronization of objects java.util.concurrent.locks.Lock

Severity	Contrast rule	Engine rule ID	Description
High	Avoid Thread Destroy	OPT.JAVA.HEB.AvoidThreadDestroy	AvoidThreadDestroy: Avoid calling java.lang.Thread.destroy
High	Avoid Static Instance Class Without Fields	OPT.JAVA.INIC.AvoidStaticInstanceClassWithoutFields	AvoidStaticInstanceClassWithoutFields: Avoid creating new objects in static initializers before a static fields have been initialized
High	A M I	OPT.JAVA.INICIA.AMI	AMI: Avoid initializations of several variables with the same value in a line
High	N F S	OPT.JAVA.INICIA.NFS	NFS: Do not use non-final static fields during the initialization
High	S F	OPT.JAVA.INICIA.SF	SF: Initialize all static fields
High	Avoid System Out Err	OPT.JAVA.IO.AvoidSystemOutErr	AvoidSystemOutErr: Use specialized library instead of System.out or System.err for logging
High	C S	OPT.JAVA.IO.CS	CS: Close input and output resources in finally blocks
High	Dont Use Print Stack Trace	OPT.JAVA.IO.DontUsePrintStackTrace	DontUsePrintStackTrace: Do not use the printStackTrace method
High	F I L B U F	OPT.JAVA.IO.FILBUF	FILBUF: All input/output resources must have a flow with buffer
High	S I E	OPT.JAVA.IO.SIE	SIE: Avoid System.err.print() or Sytem.err.println statements
High	S I O	OPT.JAVA.IO.SIO	SIO: Avoid System.out.println() or System.out.print() statements
High	Avoid Security Manager	OPT.JAVA.J2EE.AvoidSecurityManager	AvoidSecurityManager: Avoid using java.lang.SecurityManager
High	Avoid Set Property	OPT.JAVA.J2EE.AvoidSetProperty	AvoidSetProperty: Avoid calling java.security.Security.setProperty()
High	Dont Extend Class Loader	OPT.JAVA.J2SE.DontExtendClassLoader	DontExtendClassLoader: Do not allowed inheriting from java.lang.ClassLoader
High	Final Class Extends Permission	OPT.JAVA.J2SE.FinalClassExtendsPermission	FinalClassExtendsPermission: A class that extends from java.security.Permission should be final
High	Unsafe Reflection	OPT.JAVA.SEC_JAVA.UnsafeReflection	UnsafeReflection: Use of Externally-Controlled Input to Select Classes or Code ('Unsafe Reflection')
High	XPath Injection Rule	OPT.JAVA.SEC_JAVA.XPathInjectionRule	XPathInjectionRule: Improper Neutralization of Data within XPath Expressions ('XPath Injection')
High	Xslt Injection	OPT.JAVA.SEC_JAVA.XsltInjection	XsltInjection: XML Injection (aka Blind XPath Injection)
High	Use Cache With Idempotent And Safe Methods	OPT.JAVA.JAX.UseCacheWithIdempotentAndSafeMethods	UseCacheWithIdempotentAndSafeMethods: Use cache along with idempotent and safe HTTP methods
High	Validate Endpoint Business Methods	OPT.JAVA.JAX.ValidateEndpointBusinessMethods	ValidateEndpointBusinessMethods: Endpoint implementation class business methods must follow some requirements
High	Validate Endpoint Implementation Class	OPT.JAVA.JAX.ValidateEndpointImplementationClass	ValidateEndpointImplementationClass: Endpoint implementation classes must follow some requirements
High	C D B C	OPT.JAVA.JDBC.CDBC	CDBC: Close JDBC connections in finally blocks
High	U P C	OPT.JAVA.JDBC.UPC	UPC: Use a connection pool
High	Avoid Constructors Config Tests	OPT.JAVA.JUNIT_JAVA.AvoidConstructorsConfigTests	AvoidConstructorsConfigTests: Do not use constructors to set up test cases
High	C E L	OPT.JAVA.LOOP.CEL	CEL: Avoid using method calls in a loop
High	L O O P 2	OPT.JAVA.LOOP.LOOP2	LOOP2: Do not instantiate temporal Objects in loops bodies

Severity	Contrast rule	Engine rule ID	Description
High	P I	OPT.JAVA.LOOP.PI	PI: Avoid problematic constructions in a while loop
High	S Y N	OPT.JAVA.LOOP.SYN	SYN: Avoid calling synchronized methods/blocks in a loop
High	T R Y	OPT.JAVA.LOOP.TRY	TRY: Avoid using try statements in loops
High	Avoid Empty Jar Zip	OPT.JAVA.MEM.AvoidEmptyJarZip	AvoidEmptyJarZip: Avoid creating empty JAR and ZIP
High	Avoid Throw Inside Finally	OPT.JAVA.MEM.AvoidThrowInsideFinally	AvoidThrowInsideFinally: Avoid the command throws within finally
High	L E V E L	OPT.JAVA.OOP.LEVEL	LEVEL: Avoid too many levels of nested inner classes
High	A E C B	OPT.JAVA.PB.AECB	AECB: Avoid catch blocks with empty bodies
High	C L P	OPT.JAVA.PB.CLP	CLP: Avoid casting primitive data types to lower precision
High	D C F	OPT.JAVA.PB.DCF	DCF: Avoid comparing floating point types
High	D C P	OPT.JAVA.PB.DCP	DCP: Avoid using the + string concatenation operator to concatenate numbers; use it only to add numbers
High	D N I F	OPT.JAVA.PB.DNIF	DNIF: Avoid nested IF sentences with too many levels
High	D S U P	OPT.JAVA.PB.DSUP	DSUP: Place default clause at the end of a switch - case
High	E M S I	OPT.JAVA.PB.EMI	EMI: Avoid empty static blocks
High	E Q L	OPT.JAVA.PB.EQL	EQL: Use getClass() in the equals() method implementation
High	E Q L2	OPT.JAVA.PB.EQL2	EQL2: Use instanceof within an equals() method implementation
High	E S B L	OPT.JAVA.PB.ESBL	ESBL: Avoid empty synchronized blocks
High	F E B	OPT.JAVA.PB.FEB	FEB: Avoid 'for' and 'while' sentences with empty bodies
High	F L V A	OPT.JAVA.PB.FLVA	FLVA: Do not assign loop control variables in the body of a for loop
High	I E B	OPT.JAVA.PB.IEB	IEB: Avoid if statements with empty bodies
High	M A I N	OPT.JAVA.PB.MAIN	MAIN: Use the method name main() only for the entry point method
High	Don't use masterpage files C	OPT.JAVA.PB.MPC	MPC: Avoid parameter method names that provoke conflicts with class members names
High	N D C	OPT.JAVA.PB.NDC	NDC: Avoid defining direct or indirect subclasses of Error and Throwable
High	N X R E	OPT.JAVA.PB.NXRE	NXRE: Avoid defining direct or indirect subclasses of RuntimeException
High	Non Heritable Exception Classes	OPT.JAVA.PB.NonHeritableExceptionClasses	NonHeritableExceptionClasses: Do not define exceptions that inherit from the specified classes
High	P D S	OPT.JAVA.PB.PDS	PDS: Provide 'default' label for each switch statement
High	S B C	OPT.JAVA.PB.SBC	SBC: Avoid using a switch structure with a bad case statement
High	S B D F	OPT.JAVA.PB.SBDF	SBDF: Provide a break or return statement for the default label in a switch
High	U E I2	OPT.JAVA.PB.UEI2	UEI2: Use equals() when comparing Strings
High	U F S T	OPT.JAVA.PB.UFST	UFST: Avoid unconditional If blocks
High	Avoid File Separators	OPT.JAVA.PORT.AvoidFileSeparators	AvoidFileSeparators: Avoid writing directly in the code directory separators in order to create a java.io.File file

Severity	Contrast rule	Engine rule ID	Description
High	Avoid Implement Peer Interfaces	OPT.JAVA.PORT.AvoidImplementPeerInterfaces	AvoidImplementPeerInterfaces: Avoid implementing java.awt.peer interfaces
High	Avoid Call Bean From Servlet	OPT.JAVA.RECBAS.AvoidCallBeanFromServlet	AvoidCallBeanFromServlet: Avoid calling Java.beans.* from the finalize() method of a servlet
High	Avoid Call SQL From Servlet	OPT.JAVA.RECBAS.AvoidCallSQLFromServlet	AvoidCallSQLFromServlet: Avoid calling Javax.sql.* from finalize() method of a servlet
High	Avoid Get Declared Field	OPT.JAVA.REFL.AvoidGetDeclaredField	AvoidGetDeclaredField: Avoid calling java.lang.Class.getDeclaredField
High	Avoid Call Driver Manager From Servlet	OPT.JAVA.REDESC.AvoidCallDriverManagerFromServlet	AvoidCallDriverManagerFromServlet: Avoid calling java.sql.DriverManager from a servlet
High	Avoid Call Process From Servlet	OPT.JAVA.REDESC.AvoidCallProcessFromServlet	AvoidCallProcessFromServlet: Avoid calling java.lang.Process from any servlet
High	Avoid Call System From Servlet	OPT.JAVA.REDESC.AvoidCallSystemFromServlet	AvoidCallSystemFromServlet: Avoid calling java.lang.System from any servlet
High	P J D C F	OPT.JAVA.RGD.PJDCF	PJDCF: Provide Javadoc comments for public fields
High	A S F I	OPT.JAVA.RGM.ASFI	ASFI: Redecare a class with only abstract methods and static final fields as an interface
High	C T O R	OPT.JAVA.RGM.CTOR	CTOR: Avoid calling non-final, non-static and non-private methods from constructors
High	M S F	OPT.JAVA.RGM.MSF	MSF: Avoid using too many 'non-final static' fields
High	N U O T	OPT.JAVA.RGM.NUOT	NUOT: Do not use the ternary operator
High	A R L L	OPT.JAVA.RGME.ARLL	ARLL: Do not access arrays 'length' property in a loop condition
High	A G Q S	OPT.JAVA.RGOR.AGQS	AGQS: Avoid getQueryString(), using getParameter()
High	A M C O	OPT.JAVA.RGOR.AMCO	AMCO: Avoid using repeated cast over the same object or variable (Max 3 cast of every type)
High	Dont Use Reflection	OPT.JAVA.RGOR.DontUseReflection	DontUseReflection: Avoid the use of reflection
High	E I O F	OPT.JAVA.RGOR.EIOF	EIOF: Avoid if/else-if chains performing type testing
High	S D F	OPT.JAVA.RGOR.SDF	SDF: Date format class java.text.SimpleDateFormat spends many resources
High	Avoid Runtime And System Classes	OPT.JAVA.RGP.AvoidRuntimeAndSystemClasses	AvoidRuntimeAndSystemClasses: Do not use Runtime and System classes
High	Do not use applets in an application client	OPT.JAVA.RGS.NUA	NUA: Do not use applets in an application client layer
High	Transient fields in Serializable classes	OPT.JAVA.RGS.SER	SER: 'Transient' fields in 'Serializable' classes
High	Avoid EJB Explicit Server Socket	OPT.JAVA.SEC_JAVA.AvoidEJBExplicitServerSocket	AvoidEJBExplicitServerSocket: EJB Bad Practices: Use of Sockets
High	Avoid EJB Explicit Thread Management	OPT.JAVA.SEC_JAVA.AvoidEJBExplicitThreadManagement	AvoidEJBExplicitThreadManagement: Avoid explicit thread management in EJB

Severity	Contrast rule	Engine rule ID	Description
High	Avoid J2EE Jvm Exit	OPT.JAVA.SEC_JAVA.AvoidJ2EEJvmExit	AvoidJ2EEJvmExit: Avoid JVM shutdown code in J2EE applications
High	Cookies In Security Decision	OPT.JAVA.SEC_JAVA.CookiesInSecurityDecision	CookiesInSecurityDecision: Reliance on Cookies without Validation and Integrity Checking in a Security Decision
High	Database Access Control Rule	OPT.JAVA.SEC_JAVA.DatabaseAccessControlRule	DatabaseAccessControlRule: Avoid queries in the database except from the specific classes
High	J2ee File Disclosure Rule	OPT.JAVA.SEC_JAVA.J2eeFileDisclosureRule	J2eeFileDisclosureRule: File disclosure in server side J2EE forward/include
High	Not Overridable Method Rule	OPT.JAVA.SEC_JAVA.NotOverridableMethodRule	NotOverridableMethodRule: Not overridable method
High	Race Condition Format Flaw	OPT.JAVA.SEC_JAVA.RaceConditionFormatFlaw	RaceConditionFormatFlaw: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')
High	Race Condition Servlet	OPT.JAVA.SEC_JAVA.RaceConditionServlet	RaceConditionServlet: Race Condition in a Java Servlet
High	Security Check In Overridable Method Rule	OPT.JAVA.SEC_JAVA.SecurityCheckInOverridableMethodRule	SecurityCheckInOverridableMethodRule: Methods that perform a security check must be declared private or final
High	Spring No Anti Xss Configuration	OPT.JAVA.SEC_JAVA.SpringNoAntiXssConfiguration	SpringNoAntiXssConfiguration: Use defaultHtmlEscape {'OWASP-2021': ['A5'], 'WASC': ['08'], 'PCI-DSS': ['6.5.7'], 'ASVS-v4.0.2': ['3.4.5']}
High	Unhandled SSL Exception Rule	OPT.JAVA.SEC_JAVA.UnhandledSSLExceptionRule	UnhandledSSLExceptionRule: Unhandled SSL exception
High	User Controlled SQL Primary Key	OPT.JAVA.SEC_JAVA.UserControlledSQLPrimaryKey	UserControlledSQLPrimaryKey: Avoid using an user controlled Primary Key into a query
High	Avoid Call Next In Has Next	OPT.JAVA.SENT.AvoidCallNextInHasNext	AvoidCallNextInHasNext: Avoid calling next() in hasNext()
High	Avoid Call Set Size Inside Component Resized	OPT.JAVA.SENT.AvoidCallSetSizeInsideComponentResized	AvoidCallSetSizeInsideComponentResized: Avoid calling the setSize() from within componentResized()
High	Avoid Empty Sentences Switch	OPT.JAVA.SENT.AvoidEmptySentencesSwitch	AvoidEmptySentencesSwitch: Avoid empty switch statements
High	Avoid Declare Method Read Object	OPT.JAVA.SERI.AvoidDeclareMethodReadObject	AvoidDeclareMethodReadObject: Avoid declaring the method readObject() as synchronized
High	Avoid Serializable Classes	OPT.JAVA.SERI.AvoidSerializableClasses	AvoidSerializableClasses: Avoid non-serializable classes with inner classes serializable
High	Always Use Ids As Bean Identifiers	OPT.JAVA.SPRING.AlwaysUseIdsAsBeanIdentifiers	AlwaysUseIdsAsBeanIdentifiers: Use beans id attribute as beans identifiers
High	Avoid Beans With The Same Id Across Different Descriptors	OPT.JAVA.SPRING.AvoidBeansWithTheSameIdAcrossDiferentDescriptors	AvoidBeansWithTheSameIdAcrossDiferentDescriptors: Ensure beans id attribute is unique across all the XML configuration files
High	Avoid Hardcoding Null	OPT.JAVA.SPRING.AvoidHardcodingNull	AvoidHardcodingNull: Avoid initializing properties or parameters with a hardcoded null

Severity	Contrast rule	Engine rule ID	Description
High	Avoid Retrieving More Than One Batch At Time	OPT.JAVA.SPRING.AvoidRetrievingMoreThanOneBatchAtTime	AvoidRetrievingMoreThanOneBatchAtTime: Avoid retrieving more than one batch of items at time
High	Avoid Using Default Package With Auto Scanning	OPT.JAVA.SPRING.AvoidUsingDefaultPackageWithAutoScanning	AvoidUsingDefaultPackageWithAutoScanning: Avoid using the default package when using the auto scan
High	Avoid Wildcards When Loading Resource From Class Path	OPT.JAVA.SPRING.AvoidWildcardsWhenLoadingResourceFromClassPath	AvoidWildcardsWhenLoadingResourceFromClassPath: Avoid using the * wildcard when loading a resource from the classpath
High	Enable Auto Configuration Annotation Must Be Unique	OPT.JAVA.SPRING.EnableAutoConfigurationAnnotationMustBeUnique	EnableAutoConfigurationAnnotationMustBeUnique: EnableAutoConfiguration annotation must be unique
High	Refer Imported Resources To The Classpath	OPT.JAVA.SPRING.ReferImportedResourcesToTheClasspath	ReferImportedResourcesToTheClasspath: Refer imported resources to the classpath
High	Use A Proper Base Package When Using Component Scanning	OPT.JAVA.SPRING.UseAProperBasePackageWhenUsingComponentScanning	UseAProperBasePackageWhenUsingComponentScanning: Use a proper base package when using component scanning
High	Use A Top Package For Main Application Class	OPT.JAVA.SPRING.UseATopPackageForMainApplicationClass	UseATopPackageForMainApplicationClass: Main application class must belong to a root package
High	Use Constructor Based Dependency Injection	OPT.JAVA.SPRING.UseConstructorBasedDependencyInjection	UseConstructorBasedDependencyInjection: Use constructor based dependency injection
High	A C D O	OPT.JAVA.STR.ACDO	ACDO: Prevents the use of some ones of the String class constructors
High	P C T S	OPT.JAVA.STR.PCTS	PCTS: Use charAt() instead of startsWith() for comparisons of one character
High	S T O S	OPT.JAVA.STR.STOS	STOS: Avoid using toString method in a String
High	Usb In Loop	OPT.JAVA.STR.UsbInLoop	UsbInLoop: Avoid using the concatenation operator of strings +{ }
High	U S C	OPT.JAVA.STR.USC	USC: Avoid using StringBuffer to store a String that is never modified
High	Switch Fully Covers Enumeration	OPT.JAVA.SWITCH.SwitchFullyCoversEnumeration	SwitchFullyCoversEnumeration: Use a default statement and as many case statements as existing enumeration elements
High	A N F	OPT.JAVA.TRS.ANF	ANF: Avoid using notify, use notifyAll() instead
High	Avoid using variables of type java.lang.ThreadGroup	OPT.JAVA.TRS.AUTG	AUTG: Avoid using variables of type java.lang.ThreadGroup
High	M R U N	OPT.JAVA.TRS.MRUN	MRUN: Give Thread subclasses a run() method
High	N S M	OPT.JAVA.TRS.NSM	NSM: Avoid using synchronized modifier in the method declaration
High	Avoid Ejs	OPT.JAVA.WEBS.AvoidEjs	AvoidEjs: Avoid using the package com.ibm.ejs.

Severity	Contrast rule	Engine rule ID	Description
High	Avoid Ws	OPT.JAVA.WEBS.AvoidWs	AvoidWs: Avoid using Avoid using com.ibm.ws and com.ibm.websphere packages
High	Context Sensitive Data Is Kept Secure	OPT.JAVA.ANDROID.ContextSensitiveDataIsKeptSecure	ContextSensitiveDataIsKeptSecure: Avoid improper access to data created by the context
High	N IP	OPT.JAVA.DECLARA.NIP	NIP: Do not write IP address in source code
High	Avoid Exposing All Endpointl Public Methods	OPT.JAVA.JAX.AvoidExposingAllEndpointlPublicMethods	AvoidExposingAllEndpointlPublicMethods: Specify an endpoint interface to avoid exposing all the public methods
High	Information Exposure Through Error Message	OPT.JAVA.SEC_JAVA.InformationExposureThroughErrorMessage	InformationExposureThroughErrorMessage: Avoid sensitive information exposure through error messages
High	Packaged Private Key	OPT.JAVA.ANDROID.PackagedPrivateKey	PackagedPrivateKey: Avoid package private key files
High	Secure Random	OPT.JAVA.ANDROID.SecureRandom	SecureRandom: Do not use SecureRandom with a fixed seed
High	Hardcoded Salt Rule	OPT.JAVA.SEC_JAVA.HardcodedSaltRule	HardcodedSaltRule: A hardcoded salt can compromise system security
High	Inadequate Padding Rule	OPT.JAVA.SEC_JAVA.InadequatePaddingRule	InadequatePaddingRule: Inadequate padding
High	Insecure Randomness Rule	OPT.JAVA.SEC_JAVA.InsecureRandomnessRule	InsecureRandomnessRule: Standard pseudo-random number generators cannot withstand cryptographic attacks
High	Insecure Transport	OPT.JAVA.SEC_JAVA.InsecureTransport	InsecureTransport: Insecure transport
High	Insufficient Key Size Rule	OPT.JAVA.SEC_JAVA.InsufficientKeySizeRule	InsufficientKeySizeRule: Weak cryptography, insufficient key length
Info	Dont Modify Access Security	OPT.JAVA.EJB.DontModifyAccessSecurity	DontModifyAccessSecurity: Do not access or modify java.security configuration objects (Policy Security, Provider, Principal, KeyStore)
Info	Action names must end in Action	OPT.JAVA.ACTIONS.BNMC	BNMC: Action names must end in Action
Info	Use a defined package for Actions	OPT.JAVA.ACTIONS.MAUB	MAUB: Use a defined package for Actions
Info	Avoid Actions with a simple forward	OPT.JAVA.ACTIONS.OROP	OROP: Avoid Actions with a simple forward
Info	Declare unique exception for Actions	OPT.JAVA.ACTIONS.THME	THME: Declare Action methods with a unique exception type thrown
Info	Avoid Creating Unnecessary Objects	OPT.JAVA.ANDROID.AvoidCreatingUnnecessaryObjects	AvoidCreatingUnnecessaryObjects: Avoid creating string
Info	Avoid Internal Getter Setter	OPT.JAVA.ANDROID.AvoidInternalGetterSetter	AvoidInternalGetterSetter: Avoid internal use of getter/setter
Info	Private Inner Class Access	OPT.JAVA.ANDROID.PrivateInnerClassAccess	PrivateInnerClassAccess: Consider package instead of private access with private inner classes
Info	Use Value Of	OPT.JAVA.ANDROID.UseValueOf	UseValueOf: Avoid wrapper classes constructor calls
Info	E Q U A L	OPT.JAVA.BEANS.EQUAL	EQUAL: Override Object.equals() method in JavaBean classes
Info	Avoid Call To String	OPT.JAVA.CADCAR.AvoidCallToString	AvoidCallToString: Avoid calling toString () on String objects

Severity	Contrast rule	Engine rule ID	Description
Info	M D J D T	OPT.JAVA.CDCI.MDJDT	MDJDT: Use the @deprecated tag in Javadoc comments for classes and interfaces
Info	M J J D T	OPT.JAVA.CDCI.MJJDT	MJJDT: Use the @since tag in Javadoc comments for classes and interfaces
Info	M S J D T	OPT.JAVA.CDCI.MSJDT	MSJDT: Use the @see tag in Javadoc comments for classes and interfaces
Info	P J D C C4	OPT.JAVA.CDCI.PJDCC4	PJDCC4: Provide Javadoc comments for private classes and interfaces
Info	M R D C2	OPT.JAVA.CDM.MRDC2	MRDC2: Use the @return tag in Javadoc comments for protected methods
Info	M R D C4	OPT.JAVA.CDM.MRDC4	MRDC4: Use the @return tag in Javadoc comments for private methods
Info	P A R A M2	OPT.JAVA.CDM.PARAM2	PARAM2: Use the @param tag in Javadoc comments for each parameter of protected methods
Info	P A R A M4	OPT.JAVA.CDM.PARAM4	PARAM4: Use the @param tag in Javadoc comments for each parameter of private methods
Info	T H R O W2	OPT.JAVA.CDM.THROW2	THROW2: Use the @throws or @exception tags in Javadoc comments of protected methods
Info	T H R O W4	OPT.JAVA.CDM.THROW4	THROW4: Use the @throws or @exception tags in Javadoc comments of private methods
Info	C R T E	OPT.JAVA.CFFEJB.CRTE	CRTE: Declare ejbCreate() methods public, but neither static nor final
Info	F N D M	OPT.JAVA.CFFEJB.FNDM	FNDM: Declare finder methods public and neither final nor static
Info	P C R T E	OPT.JAVA.CFFEJB.PCRTE	PCRTE: Declare public 'ejbPostCreate()' method but never static nor final
Info	A L B L	OPT.JAVA.CNOM.ALBL	ALBL: Justify the opening and closing braces in code blocks
Info	E L E M	OPT.JAVA.CNOM.ELEM	ELEM: All elements (variables, methods and constructors) in a Java type should follow a naming standard
Info	I F V	OPT.JAVA.CNOM.IFV	IFV: Use all uppercase letters for the names of fields in an interface
Info	N N M A	OPT.JAVA.CNOM.NNMA	NNMA: Follow a naming standard for Java packages
Info	Avoid Short If Else	OPT.JAVA.COND.AvoidShortIfElse	AvoidShortIfElse: Avoid using if else statements in short conditions
Info	Avoid Unnecessary Constructor	OPT.JAVA.CONST.AvoidUnnecessaryConstructor	AvoidUnnecessaryConstructor: Avoid unnecessary constructors
Info	Avoid Local Variables Differ Upper Lower Case	OPT.JAVA.CONV.AvoidLocalVariablesDifferUpperLowerCase	AvoidLocalVariablesDifferUpperLowerCase: Avoid variable names that differ in only the case
Info	Avoid Same Class Field Names	OPT.JAVA.CONV.AvoidSameClassFieldNames	AvoidSameClassFieldNames: Avoid classes and fields with the same name
Info	Avoid Short Class Names	OPT.JAVA.CONV.AvoidShortClassNames	AvoidShortClassNames: Avoid class names that are less than 5 characters
Info	Use Correct Format Inner Classes	OPT.JAVA.CONV.UseCorrectFormatInnerClasses	UseCorrectFormatInnerClasses: Avoid incorrect name format of inner classes
Info	Use Is If Return Boolean	OPT.JAVA.CONV.UselsIfReturnBoolean	UselsIfReturnBoolean: Only add the prefix 'is' to method name if it returns a Boolean
Info	Avoid Declare Fields Only Null	OPT.JAVA.DECL.AvoidDeclareFieldsOnlyNull	AvoidDeclareFieldsOnlyNull: Avoid declaring fields that are only ever null

Severity	Contrast rule	Engine rule ID	Description
Info	Avoid Declare Method Final And Static	OPT.JAVA.DECL.AvoidDeclareMethodFinalAndStatic	AvoidDeclareMethodFinalAndStatic: Do not declare static methods as final
Info	Avoid Declare Method Private And Final	OPT.JAVA.DECL.AvoidDeclareMethodPrivateAndFinal	AvoidDeclareMethodPrivateAndFinal: Avoid declaring private methods as final
Info	Dont Extend Object	OPT.JAVA.DECL.DontExtendObject	DontExtendObject: Make sure that classes do not explicitly inherit 'java.lang.Object'
Info	Signature Standard To String	OPT.JAVA.DECL.SignatureStandardToString	SignatureStandardToString: Always declare the toString() with the standard signature
Info	C H A I N	OPT.JAVA.DECLARA.CHAIN	CHAIN: With multiple (overloaded) constructors, use 'this' to call more generic constructors inside constructors
Info	I Don't use masterpage files T2	OPT.JAVA.DECLARA.IMPT2	IMPT2: Use wild cards when importing a class
Info	Leave A White Space Between Arguments	OPT.JAVA.DECLARA.LeaveAWhiteSpaceBetweenArguments	LeaveAWhiteSpaceBetweenArguments: Leave a white space between arguments
Info	O C C	OPT.JAVA.DECLARA.OCC	OCC: Checks code organization inside a .java file
Info	Avoid Incorrect Format Of Final Fields	OPT.JAVA.DEN.AvoidIncorrectFormatOfFinalFields	AvoidIncorrectFormatOfFinalFields: Avoid incorrect name format in the final static fields
Info	Avoid Incorrect Format Of Final Non Static Fields	OPT.JAVA.DEN.AvoidIncorrectFormatOfFinalNonStaticFields	AvoidIncorrectFormatOfFinalNonStaticFields: Avoid incorrect name format in static non-final fields
Info	Avoid Incorrect Format Of Local Vars	OPT.JAVA.DEN.AvoidIncorrectFormatOfLocalVars	AvoidIncorrectFormatOfLocalVars: Avoid incorrect name format in local variables
Info	Avoid Incorrect Format Of Non Static Fields	OPT.JAVA.DEN.AvoidIncorrectFormatOfNonStaticFields	AvoidIncorrectFormatOfNonStaticFields: Avoid incorrect name format in the non-static fields
Info	Avoid Use Of Dollar In Names	OPT.JAVA.DEN.AvoidUseOfDollarInNames	AvoidUseOfDollarInNames: Avoid using dollar signs in the names of classes, methods or variables
Info	Interfaces Start With Capital	OPT.JAVA.DEN.InterfacesStartWithCapital	InterfacesStartWithCapital: Always start an interface name with a capital letter
Info	Method Not Start With Lowercase Or Under Score	OPT.JAVA.DEN.MethodNotStartWithLowercaseOrUnderScore	MethodNotStartWithLowercaseOrUnderScore: Always follow the java method naming conventions
Info	Start Class Name Capital	OPT.JAVA.DEN.StartClassNameCapital	StartClassNameCapital: Always start a class name with a capital letter
Info	Avoid Load Library EJB	OPT.JAVA.EJB.AvoidLoadLibraryEJB	AvoidLoadLibraryEJB: Do not load native libraries in a class of Enterprise JavaBean
Info	Declare Static Final Class EJB	OPT.JAVA.EJB.DeclareStaticFinalClassEJB	DeclareStaticFinalClassEJB: Always declare the EntityBeans static fields as final
Info	Ejb Create Enterprise Java Bean	OPT.JAVA.EJB.EjbCreateEnterpriseJavaBean	EjbCreateEnterpriseJavaBean: Always implements at least one ejbCreate () method in a class of Enterprise JavaBean
Info	Ejb Create No Parameter	OPT.JAVA.EJB.EjbCreateNoParameter	EjbCreateNoParameter: Do not have parameters for the method ejbCreate() of the MessageDrivenBean classes

Severity	Contrast rule	Engine rule ID	Description
Info	Ejb Create Post Create	OPT.JAVA.EJB.EjbCreatePostCreate	EjbCreatePostCreate: EntityBeans should one ejbPostCreate() method per ejbCreate() method
Info	Ejb Post Create Entity Bean	OPT.JAVA.EJB.EjbPostCreateEntityBean	EjbPostCreateEntityBean: EntityBeans must implement at least one method ejbPostCreate()
Info	Method Void Ejb Create	OPT.JAVA.EJB.MethodVoidEjbCreate	MethodVoidEjbCreate: Avoid non-empty return type of the methods ejbCreate() in classes of SessionBean and MessageDrivenBean
Info	Method Void Ejb Post Create	OPT.JAVA.EJB.MethodVoidEjbPostCreate	MethodVoidEjbPostCreate: Avoid non-empty return type of methods ejbPostCreate() in Enterprise JavaBean classes
Info	Use Capitals For Final Fields	OPT.JAVA.EJB.UseCapitalsForFinalFields	UseCapitalsForFinalFields: Declare fields with names in capital letters as final
Info	Define Classes Exceptions As Final	OPT.JAVA.EXCP.DefineClassesExceptionsAsFinal	DefineClassesExceptionsAsFinal: Always declare user-defined exceptions as final
Info	L L	OPT.JAVA.FMETODOS.LL	LL: Too long code lines
Info	N S A B	OPT.JAVA.FMETODOS.NSAB	NSAB: Do not place statements on the same line as {opening brace
Info	S A O P	OPT.JAVA.FMETODOS.SAOP	SAOP: Enforce number of space character(s) on each side of an assignment operator using one only line
Info	S C	OPT.JAVA.FMETODOS.SC	SC: Enforce number of space character(s) after the opening parenthesis '(' in a conditional statement
Info	S M C	OPT.JAVA.FMETODOS.SMC	SMC: Place a single space character between a method name and the opening (parenthesis
Info	Avoid Extend Thread	OPT.JAVA.HEB.AvoidExtendThread	AvoidExtendThread: Avoid extending java.lang.Thread
Info	Avoid Link Operator Assign	OPT.JAVA.INIC.AvoidLinkOperatorAssign	AvoidLinkOperatorAssign: Avoid creating assignment chains
Info	Avoid Get Context	OPT.JAVA.J2EE.AvoidGetContext	AvoidGetContext: Avoid calling java.security.AccessController.getContext()
Info	Avoid Get Security Manager	OPT.JAVA.J2EE.AvoidGetSecurityManager	AvoidGetSecurityManager: Avoid calling java.lang.System.getSecurityManager()
Info	Javadoc Location	OPT.JAVA.JDOC.JavadocLocation	JavadocLocation: Avoid putting non-Javadoc comments between Javadoc and class / method declarations
Info	Javadoc Param Rule	OPT.JAVA.JDOC.JavadocParamRule	JavadocParamRule: Always provide the tag @param in the correct order for public and protected methods
Info	Javadoc Reg Exp Rule	OPT.JAVA.JDOC.JavadocRegExpRule	JavadocRegExpRule: Do not use the tag @exception in the Javadoc comments for the exceptions thrown
Info	Format Subclass Junit Test Case	OPT.JAVA.JUNIT_JAVA.FormatSubclassJUnitTestCase	FormatSubclassJUnitTestCase: Avoid incorrect name format Junit TestCase subclasses
Info	Start Test Class Name Test Suite	OPT.JAVA.JUNIT_JAVA.StartTestClassNamedTestSuite	StartTestClassNamedTestSuite: Avoid incorrect name format TestSuite classes
Info	Avoid Call Ceil With Int Converted To Double	OPT.JAVA.MAT.AvoidCallCeilWithIntConvertedToDouble	AvoidCallCeilWithIntConvertedToDouble: Avoid passing integer type to Math.ceil() / Math.floor() / Math.round()
Info	Avoid Switch Case Break Without Comment	OPT.JAVA.PB.AvoidSwitchCaseBreakWithoutComment	AvoidSwitchCaseBreakWithoutComment: Check if there is a switch with a case without break and without comments

Severity	Contrast rule	Engine rule ID	Description
Info	Comment Every Variable	OPT.JAVA.RGD.CommentEveryVariable	CommentEveryVariable: Comment every variable
Info	P J D C F4	OPT.JAVA.RGD.PJDCF4	PJDCF4: Provide Javadoc comments for private fields
Info	P J D C M2	OPT.JAVA.RGD.PJDCM2	PJDCM2: Provide Javadoc comments for protected methods
Info	P J D C M4	OPT.JAVA.RGD.PJDCM4	PJDCM4: Provide Javadoc comments for private methods
Info	Dont Use Servlets	OPT.JAVA.RGM.DontUseServlets	DontUseServlets: Do not use servlets
Info	P F L	OPT.JAVA.RGM.PFL	PFL: Use for loops instead of while
Info	S D I	OPT.JAVA.RGM.SDI	SDI: Use blank lines to separate import blocks
Info	ESAPI Banned Rule	OPT.JAVA.SEC_JAVA.ESAPIBannedRule	ESAPIBannedRule: Avoid dangerous J2EE API, use replacements from security-focused libraries (like OWASP ESAPI)
Info	Avoid Return Sentences Method Void	OPT.JAVA.SENT.AvoidReturnSentencesMethodVoid	AvoidReturnSentencesMethodVoid: Avoid unnecessary return commands in void methods
Info	Avoid Using Do While	OPT.JAVA.SENT.AvoidUsingDoWhile	AvoidUsingDoWhile: Avoid using do-while statements
Info	Version U I D Field	OPT.JAVA.SERI.serialVersionUIDField	serialVersionUIDField: Serializable classes should always have a final static serialVersionUID field
Info	Use Package Declaration	OPT.JAVA.SWITCH.UsePackageDeclaration	UsePackageDeclaration: Always use a package declaration
Info	Password In Comment Rule	OPT.JAVA.SEC_JAVA.PasswordInCommentRule	PasswordInCommentRule: Avoid hard-coded or in-comment passwords in code
Low	Avoid Field Access Strategy	OPT.HIBERNATE.AvoidFieldAccessStrategy	AvoidFieldAccessStrategy: Use accessor methods (getter/setter) for property access
Low	Collection Getter And Setter Must Return Same Object	OPT.HIBERNATE.CollectionGetterAndSetterMustReturnSameObject	CollectionGetterAndSetterMustReturnSameObject: Collection getter and setter must return same object
Low	Declare Identifier Property	OPT.HIBERNATE.DeclareIdentifierProperty	DeclareIdentifierProperty: Declare identifier property/properties on persistent classes
Low	Externalize Queries	OPT.HIBERNATE.ExternalizeQueries	ExternalizeQueries: Externalize query strings as named queries
Low	Implement Equals Hash Code In Components	OPT.HIBERNATE.ImplementEqualsHashCodeInComponents	ImplementEqualsHashCodeInComponents: Implement equals() and hashCode() for components and composite elements
Low	Select Before Update With Update Trigger	OPT.HIBERNATE.SelectBeforeUpdateWithUpdateTrigger	SelectBeforeUpdateWithUpdateTrigger: Check select-before-update attribute matches UPDATE trigger
Low	Use Named Identifier	OPT.HIBERNATE.UseNamedIdentifier	UseNamedIdentifier: You should always specify name attribute for identifiers in Hibernate
Low	Use Proper Subclass Strategy	OPT.HIBERNATE.UseProperSubclassStrategy	UseProperSubclassStrategy: Use proper subclass strategy according to the number of subclass properties
Low	Do not manually treat exceptions in Actions	OPT.JAVA.ACTIONS.AUTC	AUTC: Do not manually treat exceptions in Actions
Low	Don't use public methods in Actions	OPT.JAVA.ACTIONS.NMTP	NMTP: Do not use public methods in Actions

Severity	Contrast rule	Engine rule ID	Description
Low	Avoid Using Floating Point	OPT.JAVA.ANDROID.AvoidUsingFloatingPoint	AvoidUsingFloatingPoint: Avoid Using Floating-Point
Low	Keep XML Layout Hierarchy Flat	OPT.JAVA.ANDROID.KeepXMLLayoutHierarchyFlat	KeepXMLLayoutHierarchyFlat: Check the depth of the layout
Low	Suspicious Import	OPT.JAVA.ANDROID.SuspiciousImport	SuspiciousImport: Check android.R package is used
Low	Use Enhanced For Loop	OPT.JAVA.ANDROID.UseEnhancedForLoop	UseEnhancedForLoop: Consider using enhanced for-each loop
Low	Use Primitive For Numbers	OPT.JAVA.ANDROID.UsePrimitiveForNumbers	UsePrimitiveForNumbers: Use primitive types instead of Number objects
Low	Use Sparse Arrays	OPT.JAVA.ANDROID.UseSparseArrays	UseSparseArrays: Use SparseArray instead of a Map with Integer keys
Low	Avoid Cyclic Dependencies Between Packages	OPT.JAVA.AvoidCyclicDependenciesBetweenPackages	AvoidCyclicDependenciesBetweenPackages: Avoid cyclic dependencies between packages
Low	Avoid Assign In While	OPT.JAVA.BUC.AvoidAssignInWhile	AvoidAssignInWhile: Avoid assignments in while do-while loop condition
Low	Avoid Call Loop	OPT.JAVA.BUC.AvoidCallLoop	AvoidCallLoop: Avoid the invocations of method within conditional loop statements
Low	Avoid Continue	OPT.JAVA.BUC.AvoidContinue	AvoidContinue: Avoid using the command continue
Low	Avoid List Contains	OPT.JAVA.BUC.AvoidListContains	AvoidListContains: Avoid calling contains inside a loop
Low	Avoid Call Objects Sub String	OPT.JAVA.CADCAR.AvoidCallObjectsSubString	AvoidCallObjectsSubString: Avoid calling substring(0) on a String object
Low	Avoid Check Index Of Positive	OPT.JAVA.CADCAR.AvoidCheckIndexOfPositive	AvoidCheckIndexOfPositive: Avoid checking if String.indexOf() is a positive value
Low	Avoid Concat String	OPT.JAVA.CADCAR.AvoidConcatString	AvoidConcatString: Declare a non-constant string as a StringBuffer
Low	M A J D T	OPT.JAVA.CDCI.MAJDT	MAJDT: Use the @author tag in Javadoc comments for classes and interfaces
Low	M V J D T	OPT.JAVA.CDCI.MVJDT	MVJDT: Use the @version tag in Javadoc comments for classes and interfaces
Low	P J D C C2	OPT.JAVA.CDCI.PJDCC2	PJDCC2: Provide Javadoc comments for protected classes and interfaces
Low	M R D C	OPT.JAVA.CDM.MRDC	MRDC: Use the @return tag in Javadoc comments for public methods
Low	T S M J T	OPT.JAVA.CDM.TSMJT	TSMJT: Provide Javadoc comment for toString() methods
Low	V M C R	OPT.JAVA.CDM.VMCR	VMCR: Avoid the @return Javadoc tag in void methods
Low	B M S S	OPT.JAVA.CFFEJB.BMSS	BMSS: Serialization of fields of a 'Bean' class
Low	Declare Public Clone	OPT.JAVA.CLON.DeclarePublicClone	DeclarePublicClone: Always declare clone() as public
Low	T R E T	OPT.JAVA.CMETRICS.TRET	TRET: Follow the limit for number of return statements
Low	C V N	OPT.JAVA.CNOM.CVN	CVN: Requires that Variable names of one character must be only used for their conventional use
Low	G E T B	OPT.JAVA.CNOM.GETB	GETB: Requires that boolean getter methods names start with 'is', 'can', 'has', or 'have'
Low	L C I N	OPT.JAVA.CNOM.LCIN	LCIN: Avoid class or interface names too long
Low	N U P R	OPT.JAVA.CNOM.NUPR	NUPR: Do not use reserved words inside Java identifiers
Low	S E T A	OPT.JAVA.CNOM.SETA	SETA: Setter methods convention

Severity	Contrast rule	Engine rule ID	Description
Low	U S F	OPT.JAVA.CNOM.USF	USF: Avoid lowercase in static and final field names
Low	D I L	OPT.JAVA.CNU.DIL	DIL: Do not explicitly import the java.lang.* package
Low	Avoid Declare Many Constructors	OPT.JAVA.COMPJ.AvoidDeclareManyConstructors	AvoidDeclareManyConstructors: Avoid declaring too many constructors
Low	Avoid Null Pointer	OPT.JAVA.COND.AvoidNullPointer	AvoidNullPointer: If/else can generate NullPointerExceptions
Low	Protected Constructor In Abstract Class	OPT.JAVA.CONST.ProtectedConstructorInAbstractClass	ProtectedConstructorInAbstractClass: Only declare 'protected' constructors for abstract classes
Low	Avoid Method Name Differ Upper Lower Case	OPT.JAVA.CONV.AvoidMethodNameDifferUpperLowerCase	AvoidMethodNameDifferUpperLowerCase: Avoid method names that differ only in the case
Low	Array List Instead Of Vector	OPT.JAVA.DECL.ArrayListInsteadOfVector	ArrayListInsteadOfVector: Always use ArrayList instead of Vector
Low	Avoid Empty Classes	OPT.JAVA.DECL.AvoidEmptyClasses	AvoidEmptyClasses: Avoid empty classes or interfaces
Low	Avoid Implement Protected Final Class	OPT.JAVA.DECL.AvoidImplementProtectedFinalClass	AvoidImplementProtectedFinalClass: Avoid implementing protected fields in a final class
Low	Avoid Use Class Interface Or Abstract	OPT.JAVA.DECL.AvoidUseClassInterfaceOrAbstract	AvoidUseClassInterfaceOrAbstract: Avoid using abstract classes or interfaces in order to declare common constants
Low	Avoid Use Static No Final	OPT.JAVA.DECL.AvoidUseStaticNoFinal	AvoidUseStaticNoFinal: Avoid non-final static fields during initialisation
Low	Avoid Using Same Variable Names	OPT.JAVA.DECL.AvoidUsingSameVariableNames	AvoidUsingSameVariableNames: Avoid using variables with the same name
Low	L In Long Object Value	OPT.JAVA.DECL.LInLongObjectValue	LInLongObjectValue: Always add a suffix to numeric literals in upper case (L instead of l)
Low	A S I	OPT.JAVA.DECLARA.ASI	ASI: Make methods static if they do not use instance class members
Low	C C P A	OPT.JAVA.DECLARA.CCPA	CCPA: Each public class must be declared in a separate file
Low	C L S	OPT.JAVA.DECLARA.CLS	CLS: Define constants on the left side of comparisons
Low	C R S	OPT.JAVA.DECLARA.CRS	CRS: Place constants on the right side of comparisons
Low	Define Every Variable In A Line	OPT.JAVA.DECLARA.DefineEveryVariableInALine	DefineEveryVariableInALine: Define every variable in a line
Low	Define Privacy Fields	OPT.JAVA.DECLARA.DefinePrivacyFields	DefinePrivacyFields: Define every field private or protected
Low	I S A C F	OPT.JAVA.DECLARA.ISACF	ISACF: Avoid interfaces that only define constants
Low	Separate Declarations And Statements	OPT.JAVA.DECLARA.SeparateDeclarationsAndStatements	SeparateDeclarationsAndStatements: Separate declarations and statements
Low	U C D C	OPT.JAVA.DECLARA.UCDC	UCDC: Provide a by default private constructor in utility classes

Severity	Contrast rule	Engine rule ID	Description
Low	Avoid Incorrect Format Of Non Static Methods	OPT.JAVA.DEN.AvoidIncorrectFormatOfNonStaticMethods	AvoidIncorrectFormatOfNonStaticMethods: Avoid the incorrect naming of non-static methods
Low	Declare Bean Class Public	OPT.JAVA.EJB.DeclareBeanClassPublic	DeclareBeanClassPublic: Always declare bean classes as public
Low	Declare Method Ejb Find Public	OPT.JAVA.EJB.DeclareMethodEjbFindPublic	DeclareMethodEjbFindPublic: Always declare ejbFindXX methods as public, non static and non final
Low	Avoid Text Field	OPT.JAVA.ESUI.AvoidTextField	AvoidTextField: Avoid using java.awt.TextField
Low	Avoid Block Catch Return	OPT.JAVA.EXCP.AvoidBlockCatchReturn	AvoidBlockCatchReturn: Do not use return statements inside catch blocks
Low	Avoid Excp Catch	OPT.JAVA.EXCP.AvoidExcpCatch	AvoidExcpCatch: Do not note an exception or throw an exception in a catch block
Low	Avoid Excp Exception	OPT.JAVA.EXCP.AvoidExcpException	AvoidExcpException: Avoid capturing java.lang.Exception exceptions
Low	Avoid New Error	OPT.JAVA.EXCP.AvoidNewError	AvoidNewError: Avoid creating new instances of java.lang.Error
Low	Next Emit No Such Element Exception	OPT.JAVA.EXCP.NextEmitNoSuchElementException	NextEmitNoSuchElementException: Always make sure that the method 'next()' in an Iterator class can be cast java.util.NoSuchElementException
Low	Avoid Unnecessary Finalize	OPT.JAVA.FIN.AvoidUnnecessaryFinalize	AvoidUnnecessaryFinalize: Avoid unnecessary finalize() method
Low	A I O C	OPT.JAVA.FMETODOS.AIOC	AIOC: Do not use instanceof to distinguish between exceptions
Low	D E E M	OPT.JAVA.FMETODOS.DEEM	DEEM: Leave at least a blank line between methods
Low	M I N D	OPT.JAVA.FMETODOS.MIND	MIND: Indent code properly
Low	N C E	OPT.JAVA.FMETODOS.NCE	NCE: Avoid Exception, RuntimeException or Throwable in catch or throw statements
Low	O S P L	OPT.JAVA.FMETODOS.OSPL	OSPL: Write one statement per line
Low	S B R	OPT.JAVA.FMETODOS.SBR	SBR: Simplify code returning boolean
Low	U P	OPT.JAVA.FMETODOS.UP	UP: Avoid unnecessary parenthesis in return statements
Low	Avoid Call Interrupted Run	OPT.JAVA.HEB.AvoidCallInterruptedRun	AvoidCallInterruptedRun: Avoid calling Thread.currentThread().interrupted() on the run() method of a thread
Low	C S I	OPT.JAVA.INICIA.CSI	CSI: Use constructors than initialize all the fields in a class
Low	Avoid Too Long Resource Names	OPT.JAVA.JAX.AvoidTooLongResourceNames	AvoidTooLongResourceNames: Avoid using too long resource names
Low	Tear Down J Unit	OPT.JAVA.JUNIT_JAVA.TearDownJUnit	TearDownJUnit: Always overwrite the tearDown() method in a JUnit test case
Low	Avoid Call Next Double	OPT.JAVA.MAT.AvoidCallNextDouble	AvoidCallNextDouble: Avoid invoking nextDouble() to generate a random int
Low	Parent Class Doesnot Reference Child Classes	OPT.JAVA.ParentClassDoesnotReferenceChildClasses	ParentClassDoesnotReferenceChildClasses: Parent class does not reference any of its child classes
Low	A D E	OPT.JAVA.PB.ADE	ADE: Avoid dangling else statements
Low	Avoid Complex If	OPT.JAVA.PB.AvoidComplexIf	AvoidComplexIf: Avoid complex condition in the statements
Low	A C E C	OPT.JAVA.RGD.ACEC	ACEC: Avoid special characters in comments

Severity	Contrast rule	Engine rule ID	Description
Low	Leave A White Line Before A Comment Line	OPT.JAVA.RGD.LeaveAWhiteLineBeforeACommentLine	LeaveAWhiteLineBeforeACommentLine: Leave a white line before a comment line
Low	P J D C F2	OPT.JAVA.RGD.PJDCF2	PJDCF2: Provide Javadoc comments for protected fields
Low	P J D C M	OPT.JAVA.RGD.PJDCM	PJDCM: Provide Javadoc comments for public methods in Java classes (non-interface)
Low	P J D C M Interface	OPT.JAVA.RGD.PJDCMInterface	PJDCMInterface: Provide Javadoc comments for public methods in Java interfaces
Low	B L K E L S E	OPT.JAVA.RGM.BLKELSE	BLKELSE: Use a branches block for 'else' statements
Low	B L K I F	OPT.JAVA.RGM.BLKIF	BLKIF: Provide a branch block for 'if' statements
Low	Dont Set In Session	OPT.JAVA.RGM.DontSetInSession	DontSetInSession: Do not place objects in session scope
Low	Use Session From Restricted Packages	OPT.JAVA.RGM.UseSessionFromRestrictedPackages	UseSessionFromRestrictedPackages: Access session objects only from classes in certain packages
Low	U E Q	OPT.JAVA.RGOR.UEQ	UEQ: Avoid comparing boolean variables with true
Low	Transient fields in Serializable classes2	OPT.JAVA.RGS.SER2	SER2: Do not use interfaces that extends the java.io.Serializable interface
Low	Improper Validation Of Array Index	OPT.JAVA.SEC_JAVA.ImproperValidationOfArrayIndex	ImproperValidationOfArrayIndex: Array index coming from a non neutralized vulnerable input
Low	Avoid Assign Same Variable	OPT.JAVA.SENT.AvoidAssignSameVariable	AvoidAssignSameVariable: Avoid assigning a variable to itself
Low	Avoid Return Null	OPT.JAVA.SENT.AvoidReturnNull	AvoidReturnNull: Avoid returning null in a method declaration
Low	Avoid Transient Fileds	OPT.JAVA.SERI.AvoidTransientFiledS	AvoidTransientFiledS: Do not declare fields as transient in non-serializable classes
Low	Implement Comparable With Serializable	OPT.JAVA.SERI.ImplementComparableWithSerializable	ImplementComparableWithSerializable: Always implement java.io.Serializable if implementing java.lang.Comparable
Low	Add Comments To Configuration Files	OPT.JAVA.SPRING.AddCommentsToConfigurationFiles	AddCommentsToConfigurationFiles: Add comments to the XML configuration files
Low	Use Rest Controller Convenience Annotation	OPT.JAVA.SPRING.UseRestControllerConvenienceAnnotation	UseRestControllerConvenienceAnnotation: Use @RestController annotation in REST application
Low	Use Spring Boot Application Convenience Annotation	OPT.JAVA.SPRING.UseSpringBootApplicationConvenienceAnnotation	UseSpringBootApplicationConvenienceAnnotation: Use @SpringBootApplication annotation
Low	Avoid New String	OPT.JAVA.STR.AvoidNewString	AvoidNewString: Avoid using the 'new' operator to create Strings of literals
Low	Do not compare class objects with getName() or getSimpleName() methods C H	OPT.JAVA.STR.CMPCH	CMPCH: Do not use strings to compare characters
Low	Avoid Short Switch	OPT.JAVA.SWITCH.AvoidShortSwitch	AvoidShortSwitch: Avoid switch statements with few branches, instead use if-else

Severity	Contrast rule	Engine rule ID	Description
Low	Avoid Declare Variable Inside Loop	OPT.JAVA.VELLOC.AvoidDeclareVariableInsideLoop	AvoidDeclareVariableInsideLoop: Avoid creating or assigning a variable within a loop
Low	Avoid New Object Get Class	OPT.JAVA.VELLOC.AvoidNewObjectGetClass	AvoidNewObjectGetClass: Avoid creating a new object with the objective of invoking getClass() on it
Low	Information Exposure Through Debug Log	OPT.JAVA.SEC_JAVA.InformationExposureThroughDebugLog	InformationExposureThroughDebugLog: Avoid exposing sensible information through log
Medium	Limit Accessibility Of Sensitive Content Provider	OPT.JAVA.ANDROID.LimitAccessibilityOfSensitiveContentProvider	LimitAccessibilityOfSensitiveContentProvider: Limit the accessibility of a app's sensitive ContentProvider
Medium	Secure Resources Properly	OPT.JAVA.JAX.SecureResourcesProperly	SecureResourcesProperly: Use annotations to secure resources properly
Medium	Use HTTP Method Annotation	OPT.JAVA.JAX.UseHTTPMethodAnnotation	UseHTTPMethodAnnotation: Use a proper annotation to indicate the HTTP request methods accepted
Medium	Use Secured Transport Layer	OPT.JAVA.JAX.UseSecuredTransportLayer	UseSecuredTransportLayer: Avoid using HTTP instead of HTTPS
Medium	Plaintext Storage In A Cookie Rule	OPT.JAVA.SEC_JAVA.PlaintextStorageInACookieRule	PlaintextStorageInACookieRule: Cleartext Storage of Sensitive Information in a Cookie
Medium	Unsafe Cookie Rule	OPT.JAVA.SEC_JAVA.UnsafeCookieRule	UnsafeCookieRule: Generate server-side cookie with adequate security properties
Medium	Exported Activity	OPT.JAVA.ANDROID.ExportedActivity	ExportedActivity: Exported activity must require permissions
Medium	Exported Preference Activity	OPT.JAVA.ANDROID.ExportedPreferenceActivity	ExportedPreferenceActivity: Activities extending PreferenceActivity should not be exported
Medium	Exported Provider	OPT.JAVA.ANDROID.ExportedProvider	ExportedProvider: Exported providers must require permissions
Medium	Exported Receiver	OPT.JAVA.ANDROID.ExportedReceiver	ExportedReceiver: Exported receivers must require permissions
Medium	Exported Service	OPT.JAVA.ANDROID.ExportedService	ExportedService: Exported services must require permissions
Medium	Avoid Host Name Checks Rule	OPT.JAVA.SEC_JAVA.AvoidHostNameChecksRule	AvoidHostNameChecksRule: Avoid checks on client-side hostname, that are not reliable due to DNS poisoning
Medium	Avoid Cartesian Product	OPT.HIBERNATE.AvoidCartesianProduct	AvoidCartesianProduct: Avoid unaware cartesian products in HQL and native SQL queries
Medium	Avoid Many To Many Associations	OPT.HIBERNATE.AvoidManyToManyAssociations	AvoidManyToManyAssociations: Avoid many-to-many associations
Medium	Avoid Non Lazy Collections	OPT.HIBERNATE.AvoidNonLazyCollections	AvoidNonLazyCollections: Avoid non-lazy persistent collections
Medium	Avoid Problematic Flush Mode	OPT.HIBERNATE.AvoidProblematicFlushMode	AvoidProblematicFlushMode: Avoid setting FlushMode that could produce stale data issues
Medium	Avoid Using HQL	OPT.HIBERNATE.AvoidUsingHQL	AvoidUsingHQL: Avoid using HQL in mapping descriptors and in code
Medium	Avoid Using Native SQL	OPT.HIBERNATE.AvoidUsingNativeSQL	AvoidUsingNativeSQL: Avoid using native SQL in mapping descriptors and in code
Medium	Cross Site History Manipulation	OPT.JAVA.SEC_JAVA.CrossSiteHistoryManipulation	CrossSiteHistoryManipulation: Cross-Site History Manipulation (XSHM)

Severity	Contrast rule	Engine rule ID	Description
Medium	Configure Connection Pool	OPT.HIBERNATE.ConfigureConnectionPool	ConfigureConnectionPool: Configure Hibernate with an explicit connection pool
Medium	Never Use Persistent Arrays	OPT.HIBERNATE.NeverUsePersistentArrays	NeverUsePersistentArrays: Never use persistent arrays
Medium	One Class Per Mapping File	OPT.HIBERNATE.OneClassPerMappingFile	OneClassPerMappingFile: Use one mapping file per persistent class
Medium	Only Use Hibernate For Database Access	OPT.HIBERNATE.OnlyUseHibernateForDatabaseAccess	OnlyUseHibernateForDatabaseAccess: Only use Hibernate/JPA for database access
Medium	Override Equals Hashcode In Composites	OPT.HIBERNATE.OverrideEqualsHashcodeInComposites	OverrideEqualsHashcodeInComposites: Every class implementing a composite-* element should override equals() and hashCode()
Medium	Proper Equals Hashcode Policy	OPT.HIBERNATE.ProperEqualsHashcodePolicy	ProperEqualsHashcodePolicy: Use the proper policy when implementing equals() and hashCode() in persistent entities
Medium	Separation Of Concerns	OPT.HIBERNATE.SeparationOfConcerns	SeparationOfConcerns: Avoid domain entities using J2EE APIs
Medium	Use Interface For Collection Property	OPT.HIBERNATE.UseInterfaceForCollectionProperty	UseInterfaceForCollectionProperty: Use interface for collection-valued properties
Medium	Use Non Final Persistent Classes	OPT.HIBERNATE.UseNonFinalPersistentClasses	UseNonFinalPersistentClasses: Avoid 'final' persistent classes
Medium	Use Nullable Type For Identifier	OPT.HIBERNATE.UseNullableTypeForIdentifier	UseNullableTypeForIdentifier: Use a nullable (non-primitive) type for identifier fields in persistent classes
Medium	Use Version Instead Of Timestamp	OPT.HIBERNATE.UseVersionInsteadOfTimestamp	UseVersionInsteadOfTimestamp: Use version instead of timestamp
Medium	Avoid Absolute Layout	OPT.JAVA.ANDROID.AvoidAbsoluteLayout	AvoidAbsoluteLayout: Avoid using AbsoluteLayout
Medium	Avoid Calls After Recycle Call	OPT.JAVA.ANDROID.AvoidCallsAfterRecycleCall	AvoidCallsAfterRecycleCall: Do not call a recycled resource
Medium	Check External Storage Permission	OPT.JAVA.ANDROID.CheckExternalStoragePermission	CheckExternalStoragePermission: Check permission usage conformance (External Storage Permission)
Medium	Check Internet Permission	OPT.JAVA.ANDROID.CheckInternetPermission	CheckInternetPermission: Check permission usage conformance (Internet Permission)
Medium	Check Location Permission	OPT.JAVA.ANDROID.CheckLocationPermission	CheckLocationPermission: Check permission usage conformance (Location Permission)
Medium	Complete Transaction With Commit	OPT.JAVA.ANDROID.CompleteTransactionWithCommit	CompleteTransactionWithCommit: Transactions have to be completed calling commit () method
Medium	Format String Injection Rule	OPT.JAVA.SEC_JAVA.FormatStringInjectionRule	FormatStringInjectionRule: Exclude unsanitized user input from format strings
Medium	Do Not Log Sensitive Information	OPT.JAVA.ANDROID.DoNotLogSensitiveInformation	DoNotLogSensitiveInformation: Avoid unsafe log access
Medium	Dont Use Multidimensional List	OPT.JAVA.ANDROID.DontUseMultidimensionalList	DontUseMultidimensionalList: Avoid using multidimensional arrays or nested lists
Medium	Input Path Not Canonicalized Rule	OPT.JAVA.SEC_JAVA.InputPathNotCanonicalizedRule	InputPathNotCanonicalizedRule: Incorrect Behavior Order: Validate Before Canonicalize

Severity	Contrast rule	Engine rule ID	Description
Medium	Missing Recycle Calls	OPT.JAVA.ANDROID.MissingRecycleCalls	MissingRecycleCalls: Resources must be recycled after using them
Medium	On Click Method Does Not Exist	OPT.JAVA.ANDROID.OnClickMethodDoesNotExist	OnClickMethodDoesNotExist: onClick method must exists
Medium	Request Parameters In Session Rule	OPT.JAVA.SEC_JAVA.RequestParametersInSessionRule	RequestParametersInSessionRule: Request parameters should not be passed into Session without sanitizing
Medium	View Holder	OPT.JAVA.ANDROID.ViewHolder	ViewHolder: Avoid inflate a new layout when implementing a view Adapter
Medium	A Package Does Not Depend On Less Stable Packages	OPT.JAVA.APackageDoesNotDependOnLessStablePackages	APackageDoesNotDependOnLessStablePackages: A package does not depend on less stable packages
Medium	Do Not Return Store Mutable Members	OPT.JAVA.DoNotReturnStoreMutableMembers	DoNotReturnStoreMutableMembers: Do not directly return or store references to mutable members
Medium	Avoid Peer Interface	OPT.JAVA.AWT.AvoidPeerInterface	AvoidPeerInterface: Avoid using java.awt.peer interfaces directly
Medium	P P B	OPT.JAVA.BEANS.PPB	PPB: Avoid passing presentation layer objects to beans
Medium	Avoid Enums In While	OPT.JAVA.BUC.AvoidEnumsInWhile	AvoidEnumsInWhile: Avoid using the enumerator for loop control
Medium	Avoid For Without Ini Incr	OPT.JAVA.BUC.AvoidForWithoutIniIncr	AvoidForWithoutIniIncr: Avoid loops without an initiator and an increase
Medium	Avoid Array To String	OPT.JAVA.CADCAR.AvoidArrayToString	AvoidArrayToString: Avoid calling toString() on an array
Medium	N C C A	OPT.JAVA.CDCI.NCCA	NCCA: Old commented code
Medium	P A R A M	OPT.JAVA.CDM.PARAM	PARAM: Use the @param tag in Javadoc comments for each parameter of public methods
Medium	T H R O W	OPT.JAVA.CDM.THROW	THROW: Use the @throws or @exception tags in Javadoc comments of public methods
Medium	Use Set Rollback Only	OPT.JAVA.CFFEJB.UseSetRollbackOnly	UseSetRollbackOnly: Call setRollbackOnly() if a method in a bean throws an application exception
Medium	Classes Are Strongly Internally Coupled	OPT.JAVA.ClassesAreStronglyInternallyCoupled	ClassesAreStronglyInternallyCoupled: Classes internally strongly coupled must be avoided
Medium	Class Hierarchies Are Not Too Deep	OPT.JAVA.ClassHierarchiesAreNotTooDeep	ClassHierarchiesAreNotTooDeep: Avoid too deep hierarchy classes
Medium	Avoid Overload Clone	OPT.JAVA.CLON.AvoidOverloadClone	AvoidOverloadClone: Avoid overloading the clone method
Medium	Implement Cloneable Alter Clone	OPT.JAVA.CLON.ImplementCloneableAlterClone	ImplementCloneableAlterClone: Implement java.lang.Cloneable when overwriting clone() method
Medium	Signature Standard Clone	OPT.JAVA.CLON.SignatureStandardClone	SignatureStandardClone: Always use the standard signature for clone methods
Medium	N P R I F	OPT.JAVA.CMETRICS.NPRIF	NPRIF: Maximum allowed number of private fields
Medium	N P R I M	OPT.JAVA.CMETRICS.NPRIM	NPRIM: Maximum allowed number of private methods
Medium	N P R O F	OPT.JAVA.CMETRICS.NPROF	NPROF: Maximum allowed number of protected fields
Medium	N P R O M	OPT.JAVA.CMETRICS.NPROM	NPROM: Maximum allowed number of protected methods

Severity	Contrast rule	Engine rule ID	Description
Medium	N P U B F	OPT.JAVA.CMETRICS.NPUBF	NPUBF: Maximum allowed number of public fields
Medium	N P U B M	OPT.JAVA.CMETRICS.NPUBM	NPUBM: Maximum allowed number of public methods
Medium	S T M T	OPT.JAVA.CMETRICS.STMT	STMT: Follow the limit for number of statements in a method
Medium	T N L M	OPT.JAVA.CMETRICS.TNLM	TNLM: Follow the limit for number of lines in a method
Medium	T N M C	OPT.JAVA.CMETRICS.TNMC	TNMC: Counts the number of method calls
Medium	T N O P	OPT.JAVA.CMETRICS.TNOP	TNOP: Avoid methods with too many parameters
Medium	G E T A	OPT.JAVA.CNOM.GETA	GETA: Getter methods conventions
Medium	I R B	OPT.JAVA.CNOM.IRB	IRB: Use is... only for naming methods that return a boolean
Medium	N A R G S	OPT.JAVA.CNOM.NARGS	NARGS: Use a naming convention for arguments
Medium	N C L	OPT.JAVA.CNOM.NCL	NCL: Naming convention for class names
Medium	N E	OPT.JAVA.CNOM.NE	NE: Naming convention for class exceptions
Medium	N M	OPT.JAVA.CNOM.NM	NM: Follow a naming standard for methods
Medium	Inner classes should be protected or private	OPT.JAVA.CNOM.PKG	PKG: Use all lowercase letters for package names
Medium	R Inner classes should be protected or private	OPT.JAVA.CNOM.RPKG	RPKG: Avoid using the package name that is reserved by Sun
Medium	D I C	OPT.JAVA.COL.DIC	DIC: Define initial capacities for ArrayList, HashMap, HashSet, Hashtable, Vector and WeakHashMap
Medium	M C S	OPT.JAVA.COL.MCS	MCS: Minimize using synchronized collections: detect use of synchronized collections (Vector, Hashtable) and indicate it
Medium	M C S C	OPT.JAVA.COL.MCSC	MCSC: Do not make an excessive use of synchronized collections: detect use of Collections.synchronizedCollection() and derivate collections
Medium	Avoid Bit To Bit Comparisons	OPT.JAVA.COMP.AvoidBitToBitComparisons	AvoidBitToBitComparisons: Avoid using bitwise operators to make comparisons
Medium	Avoid Comp Byte Max Value Min Value	OPT.JAVA.COMP.AvoidCompByteMaxValueMinValue	AvoidCompByteMaxValueMinValue: Avoid the comparisons of byte data with Byte.MIN_VALUE and Byte.MAX_VALUE because they always return the same result
Medium	Avoid Comp Character Max Value Min Value	OPT.JAVA.COMP.AvoidCompCharacterMaxValueMinValue	AvoidCompCharacterMaxValueMinValue: Avoid comparisons between Character.MIN_VALUE and Character.MAX_VALUE, because they always return the same result
Medium	Avoid Comp Double Max Value Min Value	OPT.JAVA.COMP.AvoidCompDoubleMaxValueMinValue	AvoidCompDoubleMaxValueMinValue: Avoid comparing double with Double.MAX_VALUE and Double.MIN_VALUE. These comparisons will always return the same value
Medium	Avoid Comp Float Max Value Min Value	OPT.JAVA.COMP.AvoidCompFloatMaxValueMinValue	AvoidCompFloatMaxValueMinValue: Avoid comparing Float.MAX_VALUE and Double.MIN_VALUE. These comparisons will always return the same value
Medium	Avoid Comp Integer Max Value Min Value	OPT.JAVA.COMP.AvoidCompIntegerMaxValueMinValue	AvoidCompIntegerMaxValueMinValue: Avoid the comparisons of Integer.MIN_VALUE and Integer.MAX_VALUE because they always return the same result

Severity	Contrast rule	Engine rule ID	Description
Medium	Avoid Comp Long Max Value Min Value	OPT.JAVA.COMP.AvoidCompLongMaxValueMinValue	AvoidCompLongMaxValueMinValue: Avoid comparisons with Long.MIN_VALUE and Long.MAX_VALUE, because they always return the same result
Medium	Avoid Comp Short Max Value Min Value	OPT.JAVA.COMP.AvoidCompShortMaxValueMinValue	AvoidCompShortMaxValueMinValue: Avoid comparing short with Short.MAX_VALUE and Short.MIN_VALUE. These comparisons will always return the same value
Medium	Dont Compare One Self	OPT.JAVA.COMP.DontCompareOneSelf	DontCompareOneSelf: Avoid comparing a variable with itself
Medium	Avoid Excessively Bracketed Loops	OPT.JAVA.COMPJ.AvoidExcessivelyBracketedLoops	AvoidExcessivelyBracketedLoops: Avoid nesting more than three loops
Medium	Avoidt Nest Try Catch	OPT.JAVA.COMPJ.AvoidtNestTryCatch	AvoidtNestTryCatch: Avoid nesting more than 1 try / catch
Medium	Components Are Not Calling Too Many Other Components	OPT.JAVA.ComponentsAreNotCallingTooManyOtherComponents	ComponentsAreNotCallingTooManyOtherComponents: Avoid using components calling too many other components
Medium	Avoid Assignments Within I F	OPT.JAVA.COND.AvoidAssignmentsWithinIF	AvoidAssignmentsWithinIF: Avoid assignments inside conditional expressions
Medium	Private Constructor Utility Program	OPT.JAVA.CONST.PrivateConstructorUtilityProgram	PrivateConstructorUtilityProgram: Only declare 'private' constructors for a class of a utility program
Medium	Avoid Field Variables Differ Upper Lower Case	OPT.JAVA.CONV.AvoidFieldVariablesDifferUpperLowerCase	AvoidFieldVariablesDifferUpperLowerCase: Avoid field names that differ only in case
Medium	Avoid Import Class Local Package	OPT.JAVA.CONV.AvoidImportClassLocalPackage	AvoidImportClassLocalPackage: Avoid importing classes from the local package
Medium	Avoid Parameter Differ Upper Lower Case	OPT.JAVA.CONV.AvoidParameterDifferUpperLowerCase	AvoidParameterDifferUpperLowerCase: Avoid parameters that only differ in case
Medium	Avoid Same Method Field Names	OPT.JAVA.CONV.AvoidSameMethodFieldNames	AvoidSameMethodFieldNames: Avoid fields and methods with the same name
Medium	Collection Type Verification	OPT.JAVA.CONV.CollectionTypeVerification	CollectionTypeVerification: Always test the type of object when it is converted to an abstract collection
Medium	Object Type Verification	OPT.JAVA.CONV.ObjectTypeVerification	ObjectTypeVerification: Always check object type before cast
Medium	Avoid Declare Field Used In Only One Method	OPT.JAVA.DECL.AvoidDeclareFieldUsedInOnlyOneMethod	AvoidDeclareFieldUsedInOnlyOneMethod: Do not declare private fields that are used in only one method
Medium	Avoid Final Static Public Matrix Field	OPT.JAVA.DECL.AvoidFinalStaticPublicMatrixField	AvoidFinalStaticPublicMatrixField: Avoid using a public static final field of type Matrix
Medium	Avoid Local Variable Similar Name	OPT.JAVA.DECL.AvoidLocalVariableSimilarName	AvoidLocalVariableSimilarName: Avoid local variables with similar names
Medium	Avoid New Boolean	OPT.JAVA.DECL.AvoidNewBoolean	AvoidNewBoolean: Avoid creating Boolean objects using Boolean.TRUE or Boolean.FALSE
Medium	Avoid Num Literals	OPT.JAVA.DECL.AvoidNumLiterals	AvoidNumLiterals: Avoid using numeric literals
Medium	Avoid Use Literal	OPT.JAVA.DECL.AvoidUseLiteral	AvoidUseLiteral: Avoid using explicit string literals, but instead, declare constants

Severity	Contrast rule	Engine rule ID	Description
Medium	Private Constructor In Final Class	OPT.JAVA.DECL.PrivateConstructorInFinalClass	PrivateConstructorInFinalClass: Declare classes where all constructors are private as 'final'
Medium	A U V T	OPT.JAVA.DECLARA.AUVT	AUVT: Declare the List and Set variables with their interface type
Medium	D C I	OPT.JAVA.DECLARA.DCI	DCI: Define public constants in an interface
Medium	D C T O R	OPT.JAVA.DECLARA.DCTOR	DCTOR: Define a default constructor whenever possible
Medium	I Don't use masterpage files T	OPT.JAVA.DECLARA.IMPT	IMPT: Avoid usage of * in import statements
Medium	M V O S	OPT.JAVA.DECLARA.MVOS	MVOS: Avoid declaring multiple variables in one statement
Medium	N T X	OPT.JAVA.DECLARA.NTX	NTX: Avoid throwing 'Exception'. Always use a proper Exception subclass
Medium	Not Define Object Variables	OPT.JAVA.DECLARA.NotDefineObjectVariables	NotDefineObjectVariables: Do not define 'Object' variables
Medium	Avoid Abstract Class Ejb	OPT.JAVA.EJB.AvoidAbstractClassEjb	AvoidAbstractClassEjb: Do not declare abstract EJBs
Medium	Avoid Create Exception	OPT.JAVA.EJB.AvoidCreateException	AvoidCreateException: Always launch 'javax.ejb.CreateException' for creation methods of the remote and local interfaces
Medium	Avoid Final Class Ejb	OPT.JAVA.EJB.AvoidFinalClassEjb	AvoidFinalClassEjb: Do not declare EJBs as final
Medium	Avoid Finder Exception	OPT.JAVA.EJB.AvoidFinderException	AvoidFinderException: Always throw a 'javax.ejb.FinderException' for the finder methods of the remote and local interfaces
Medium	Avoid Return This Class Ejb	OPT.JAVA.EJB.AvoidReturnThisClassEjb	AvoidReturnThisClassEjb: Avoid returning 'this' from the methods of the Enterprise JavaBean classes
Medium	Avoid Using Thread	OPT.JAVA.EJB.AvoidUsingThread	AvoidUsingThread: Do not start, stop or manage objects in a EntityBeans java.lang.Thread
Medium	Declareejb Post Create Public	OPT.JAVA.EJB.DeclareejbPostCreatePublic	DeclareejbPostCreatePublic: Always declare ejbPostCreate() public
Medium	Unnormalized Input String	OPT.JAVA.SEC_JAVA.UnnormalizedInputString	UnnormalizedInputString: Always normalize system inputs
Medium	Dont Pass This As Arg	OPT.JAVA.EJB.DontPassThisAsArg	DontPassThisAsArg: Do not pass 'this' as an argument
Medium	Ejb Create Message Driven Bean	OPT.JAVA.EJB.EjbCreateMessageDrivenBean	EjbCreateMessageDrivenBean: Always implement the least one ejbCreate() method in a class of MessageDrivenBean
Medium	Avoid Text Area	OPT.JAVA.ESUI.AvoidTextArea	AvoidTextArea: Avoid using java.awt.TextArea
Medium	Avoid Empty Finally Blocks	OPT.JAVA.EXCP.AvoidEmptyFinallyBlocks	AvoidEmptyFinallyBlocks: Avoid empty final blocks
Medium	Avoid Empty Try Blocks	OPT.JAVA.EXCP.AvoidEmptyTryBlocks	AvoidEmptyTryBlocks: Avoid empty try blocks
Medium	Avoid Excp Error	OPT.JAVA.EXCP.AvoidExcpError	AvoidExcpError: Avoid java.lang.Error catch exceptions
Medium	Avoid Illegal Monitor State Excp	OPT.JAVA.EXCP.AvoidIllegalMonitorStateExcp	AvoidIllegalMonitorStateExcp: Avoid capturing java.lang.IllegalMonitorStateException
Medium	Avoid New Exception	OPT.JAVA.EXCP.AvoidNewException	AvoidNewException: Avoid creating new instances of java.lang.Exception
Medium	Avoid Object Throwable	OPT.JAVA.EXCP.AvoidObjectThrowable	AvoidObjectThrowable: Avoid capturing Throwable objects
Medium	Exceptions In Throws	OPT.JAVA.EXCP.ExceptionsInThrows	ExceptionsInThrows: Always use specific exceptions in the throws clause
Medium	A M C	OPT.JAVA.FMETODOS.AMC	AMC: Avoid multiple comparisons within if

Severity	Contrast rule	Engine rule ID	Description
Medium	D U N	OPT.JAVA.FMETODOS.DUN	DUN: Too many uses of the negation operator (!) in a method
Medium	G N E	OPT.JAVA.FMETODOS.GNE	GNE: Control number of exceptions in a clause
Medium	I A D	OPT.JAVA.FMETODOS.IAD	IAD: Declare arrays with [] braces after the array type and before the variable name(s)
Medium	Avoid Call Start Constructors	OPT.JAVA.HEB.AvoidCallStartConstructors	AvoidCallStartConstructors: Do not call the start() method from within the constructors of the subclass of the thread
Medium	Avoid This In Sincronized Sentences	OPT.JAVA.HEB.AvoidThisInSincronizedSentences	AvoidThisInSincronizedSentences: Avoid using this in synchronised blocks
Medium	Avoid Wait Outside Brackets	OPT.JAVA.HEB.AvoidWaitOutsideBrackets	AvoidWaitOutsideBrackets: Avoid using wait() method of the class java.lang.Object outside a while loop
Medium	D V B C	OPT.JAVA.INICIA.DVBC	DVBC: Define class attributes at the beginning of the class
Medium	L V	OPT.JAVA.INICIA.LV	LV: Initialize all local variables at the declaration statement
Medium	U S N	OPT.JAVA.INICIA.USN	USN: Avoid literals in method calls. Named constants (static final class variables) make source code easier to understand and maintain
Medium	Avoid Check Permission	OPT.JAVA.J2EE.AvoidCheckPermission	AvoidCheckPermission: Avoid calling java.security.AccessController.checkPermission()
Medium	Avoid Class Loader Instance	OPT.JAVA.J2EE.AvoidClassLoaderInstance	AvoidClassLoaderInstance: Avoid instantiating java.lang.ClassLoader
Medium	Avoid Do As	OPT.JAVA.J2EE.AvoidDoAs	AvoidDoAs: Avoid calling javax.security.auth.Subject.doAs()
Medium	Avoid Do As Privileged	OPT.JAVA.J2EE.AvoidDoAsPrivileged	AvoidDoAsPrivileged: Avoid calling javax.security.auth.Subject.doAsPrivileged()
Medium	Avoid Do Privileged	OPT.JAVA.J2EE.AvoidDoPrivileged	AvoidDoPrivileged: Avoid calling java.security.AccessController.doPrivileged()
Medium	Avoid Extend Permission	OPT.JAVA.J2EE.AvoidExtendPermission	AvoidExtendPermission: Avoid extending java.security.Permission
Medium	Avoid Get Policy	OPT.JAVA.J2EE.AvoidGetPolicy	AvoidGetPolicy: Avoid calling Policy.getPolicy()
Medium	Avoid Get Property	OPT.JAVA.J2EE.AvoidGetProperty	AvoidGetProperty: Avoid calling java.security.Security.getProperty()
Medium	Avoid Large Blocks	OPT.JAVA.J2EE.AvoidLargeBlocks	AvoidLargeBlocks: Avoiding a privileged block of more than 5 lines
Medium	Avoid Permission Instance	OPT.JAVA.J2EE.AvoidPermissionInstance	AvoidPermissionInstance: Avoid creating an instance of java.security.Permission
Medium	Avoid Policy Instance	OPT.JAVA.J2EE.AvoidPolicyInstance	AvoidPolicyInstance: Avoid creating an instance of java.security.Policy
Medium	Avoid Set Policy	OPT.JAVA.J2EE.AvoidSetPolicy	AvoidSetPolicy: Avoid call to Policy.setPolicy()
Medium	Avoid Alter Check Security Manager	OPT.JAVA.J2SE.AvoidAlterCheckSecurityManager	AvoidAlterCheckSecurityManager: Avoid overwriting methods check* in SecurityManager
Medium	Avoid Catch Security Exception	OPT.JAVA.J2SE.AvoidCatchSecurityException	AvoidCatchSecurityException: Avoid capturing Java.lang.SecurityException
Medium	Avoid Static Public No Final Field	OPT.JAVA.J2SE.AvoidStaticPublicNoFinalField	AvoidStaticPublicNoFinalField: Avoid non-final public static fields
Medium	Avoid Using J A X R P C	OPT.JAVA.JAX.AvoidUsingJAXRPC	AvoidUsingJAXRPC: Avoid using JAX-RPC

Severity	Contrast rule	Engine rule ID	Description
Medium	Include A P I Version Number Into URL	OPT.JAVA.JAX.IncludeAPIVersionNumberIntoURL	IncludeAPIVersionNumberIntoURL: Include the API version number into the URL
Medium	Use Default Value Annotation	OPT.JAVA.JAX.UseDefaultValueAnnotation	UseDefaultValueAnnotation: Use the @DefaultValue annotation when processing a request
Medium	Use Generic Entity	OPT.JAVA.JAX.UseGenericEntity	UseGenericEntity: Use GenericEntity when returning a list of instances
Medium	Use JSON Responses	OPT.JAVA.JAX.UseJSONResponses	UseJSONResponses: Use JSON responses in REST applications
Medium	Use Proper Return Types And Response Codes	OPT.JAVA.JAX.UseProperReturnTypesAndResponseCodes	UseProperReturnTypesAndResponseCodes: Include a proper response code along with an appropriate return type
Medium	P S T	OPT.JAVA.JDBC.PST	PST: Use PreparedStatement instead of Statement to similar requests
Medium	R R W D	OPT.JAVA.JDBC.RRWD	RRWD: Close JDBC resources when finishing using
Medium	S E L E C T	OPT.JAVA.JDBC.SELECT	SELECT: Avoid SQL sentences with SELECT *
Medium	Javadoc T O D O	OPT.JAVA.JDOC.JavadocTODO	JavadocTODO: Avoid TODO comments in production code
Medium	Javadoc Throws Rule	OPT.JAVA.JDOC.JavadocThrowsRule	JavadocThrowsRule: Provide the @throws tag in Javadoc comments for public and protected methods
Medium	Call Super Set Up From Set Up	OPT.JAVA.JUNIT_JAVA.CallSuperSetUpFromSetUp	CallSuperSetUpFromSetUp: Always call super.setUp() from the method setUp of the JUnit TestCase
Medium	Call Super Tear Down From Tear Down	OPT.JAVA.JUNIT_JAVA.CallSuperTearDownFromTearDown	CallSuperTearDownFromTearDown: Always call super.tearDown() from the method tearDown() of the JUnit TestCase
Medium	Invoke Suite As Static Public	OPT.JAVA.JUNIT_JAVA.InvokeSuiteAsStaticPublic	InvokeSuiteAsStaticPublic: Always declare Junit.framework.TestCase.suite() as a public static method
Medium	I A C B	OPT.JAVA.LOOP.IACB	IACB: Use System.arraycopy() instead of using a loop to copy arrays
Medium	L I N T	OPT.JAVA.LOOP.LINT	LINT: Use integer index variable in loops
Medium	Avoid Call Run From Servlet	OPT.JAVA.MAN.AvoidCallRunFromServlet	AvoidCallRunFromServlet: Avoid calling java.lang.Runnable.run() from any servlet
Medium	Avoid Confused Multiplication	OPT.JAVA.MAT.AvoidConfusedMultiplication	AvoidConfusedMultiplication: Avoid Confusing Multiplications
Medium	Avoid Modulus One	OPT.JAVA.MAT.AvoidModulusOne	AvoidModulusOne: Avoid modulus value that equal to one
Medium	Avoid Static Call Math	OPT.JAVA.MAT.AvoidStaticCallMath	AvoidStaticCallMath: Avoid invoking a static method of java.lang.Math invoke on a constant
Medium	Check Variable Odd Value	OPT.JAVA.MAT.CheckVariableOddValue	CheckVariableOddValue: Avoid using var% 2 != 0
Medium	Avoid Number Instance With New	OPT.JAVA.MEM.AvoidNumberInstanceWithNew	AvoidNumberInstanceWithNew: Avoid creating a instance of a number using new
Medium	Avoid Run Finalization	OPT.JAVA.MEM.AvoidRunFinalization	AvoidRunFinalization: Avoid call to java.lang.Runtime.runFinalization()
Medium	Avoid Return Null Enumeration	OPT.JAVA.NULL.AvoidReturnNullEnumeration	AvoidReturnNullEnumeration: Return empty enumerations instead of returning null
Medium	Avoid Return Null Iterator	OPT.JAVA.NULL.AvoidReturnNullIterator	AvoidReturnNullIterator: Return an empty iterator instead of null

Severity	Contrast rule	Engine rule ID	Description
Medium	Null Dereference	OPT.JAVA.NullDereference	NullDereference: NULL Pointer Dereference
Medium	D N C S S	OPT.JAVA.PB.DNCSS	DNCSS: Do not call setSize() in ComponentListener.componentResized()
Medium	Empty pages	OPT.JAVA.PB.ISEM	ISEM: Avoid calling varString.equals("literal") or varString.equalsIgnoreCase("literal")
Medium	J U I N	OPT.JAVA.PB.JUIN	JUIN: Avoid incrementers with error
Medium	M A S P	OPT.JAVA.PB.MASP	MASP: Assign protected accessibility to readResolve () and writeReplace () methods in serializable classes
Medium	MVC	OPT.JAVA.PB.MVC	MVC: Avoid local variables with the same name as variables of the enclosing class
Medium	Num Max Class By Package	OPT.JAVA.PB.NumMaxClassByPackage	NumMaxClassByPackage: Avoid an excessive number of classes per package/namespace
Medium	R F F B	OPT.JAVA.PB.RFFB	RFFB: Return from finally blocks
Medium	U E I	OPT.JAVA.PB.UEI	UEI: Use equals() when comparing Objects
Medium	U O M E	OPT.JAVA.PB.UOME	UOME: Do not make useless method overwrites
Medium	Avoid Dump Stack	OPT.JAVA.PERF.AvoidDumpStack	AvoidDumpStack: Avoid using Thread.dumpStack() in production code
Medium	Avoid System	OPT.JAVA.PERF.AvoidSystem	AvoidSystem: Avoid using System.out or System.err in production code
Medium	Avoid Call Connection From Servlet	OPT.JAVA.RECBAS.AvoidCallConnectionFromServlet	AvoidCallConnectionFromServlet: Avoid calling java.sql.Connection from the finalize() method of a servlet
Medium	Avoid Call Net From Servlet	OPT.JAVA.RECBAS.AvoidCallNetFromServlet	AvoidCallNetFromServlet: Avoid calling Java.net .* from finalize() method of a servlet
Medium	Avoid Call Omg From Servlet	OPT.JAVA.RECBAS.AvoidCallOmgFromServlet	AvoidCallOmgFromServlet: Avoid calling org.omg .* from finalize() method of a servlet
Medium	Avoid Call Result Set From Servlet	OPT.JAVA.RECBAS.AvoidCallResultSetFromServlet	AvoidCallResultSetFromServlet: Avoid calling java.sql.ResultSet from the finalize() method of a servlet
Medium	Avoid Call Statement From Servlet	OPT.JAVA.RECBAS.AvoidCallStatementFromServlet	AvoidCallStatementFromServlet: Avoid calling java.sql.Statement from the method finalize() of a servlet
Medium	Avoid Date Matrix	OPT.JAVA.RECBASJ2SE.AvoidDateMatrix	AvoidDateMatrix: Avoid using a matrix of type Date
Medium	Avoid Call Jar From Servlet	OPT.JAVA.RENDESC.AvoidCallJarFromServlet	AvoidCallJarFromServlet: Avoid calling Java.util.jar .* from any servlet
Medium	Avoid Call Rmi From Servlet	OPT.JAVA.RENDESC.AvoidCallRmiFromServlet	AvoidCallRmiFromServlet: Avoid calling Java.rmi .* from any servlet
Medium	A D G C	OPT.JAVA.RGM.ADGC	ADGC: Avoid using java.util.Date or java.util.GregorianCalendar
Medium	A F P	OPT.JAVA.RGM.AFP	AFP: Avoid modifications on method or constructor parameters
Medium	A R N	OPT.JAVA.RGM.ARN	ARN: Return zero-length arrays instead of null
Medium	B L K D O W H L	OPT.JAVA.RGM.BLKDOWNHL	BLKDOWNHL: Use a braces block for 'do-while' statements
Medium	B L K F O R	OPT.JAVA.RGM.BLKFOR	BLKFOR: Use a braces block for 'for' statements
Medium	B L K W H L	OPT.JAVA.RGM.BLKWHL	BLKWHL: Provide a braces block for 'while' statements
Medium	C L N C	OPT.JAVA.RGM.CLNC	CLNC: Avoid using constructors in the clone() method
Medium	Make your clone() method final for security	OPT.JAVA.RGM.CLONE	CLONE: Call super.clone() in all clone() methods
Medium	D U I D	OPT.JAVA.RGM.DUID	DUID: Create a serialVersionUID for all Serializable classes

Severity	Contrast rule	Engine rule ID	Description
Medium	Dont Use Assert	OPT.JAVA.RGM.DontUseAssert	DontUseAssert: Do not use assert and do not launch AssertionError
Medium	F F	OPT.JAVA.RGM.FF	FF: Declare constant fields private and final
Medium	F L V	OPT.JAVA.RGM.FLV	FLV: Declare constant local variables as final
Medium	H M F	OPT.JAVA.RGM.HMF	HMF: Avoid giving method local variables and parameters the same name as class fields
Medium	P C I F	OPT.JAVA.RGM.PCIF	PCIF: Declare for loops with a condition and an increment statement
Medium	P C I F2	OPT.JAVA.RGM.PCIF2	PCIF2: Use only one condition statement in for blocks
Medium	P C I F3	OPT.JAVA.RGM.PCIF3	PCIF3: Use 'for' loops with explicit condition and increment/decrement statements
Medium	P C T O R	OPT.JAVA.RGM.PCTOR	PCTOR: Do not declare public constructors in non-public classes
Medium	P S F A	OPT.JAVA.RGM.PSFA	PSFA: Avoid using public static final array fields
Medium	U S T	OPT.JAVA.RGM.UST	UST: Use StringTokenizer instead of indexOf() or substring() for String parsing
Medium	C I P A	OPT.JAVA.RGME.CIPA	CIPA: Avoid large array initializations from constant arrays
Medium	O O M E	OPT.JAVA.RGME.OOME	OOME: Capture OutOfMemoryError for large arrays initializations
Medium	U N C	OPT.JAVA.RGOR.UNC	UNC: Do not cast a variable to an interface implemented by that variable or to the base class that extends it
Medium	Use Service Locator Pattern	OPT.JAVA.RGOR.UseServiceLocatorPattern	UseServiceLocatorPattern: Apply the 'Service Locator' pattern
Medium	Avoid Packages	OPT.JAVA.RGP.AvoidPackages	AvoidPackages: Do not import the specified packages
Medium	L N S P	OPT.JAVA.RGP.LNSP	LNSP: Do not hard code character for line separator
Medium	No Absolute Paths	OPT.JAVA.RGP.NoAbsolutePaths	NoAbsolutePaths: Do not use absolute paths
Medium	Make your clone() method final for security	OPT.JAVA.RGS.CLONE	CLONE: Make your clone() method final for security
Medium	Do not compare class objects with getName() or getSimpleName() methods	OPT.JAVA.RGS.CMP	CMP: Do not compare class objects with getName() or getSimpleName() methods
Medium	Define inner classes as private	OPT.JAVA.RGS.INNER	INNER: Define inner classes as private
Medium	Inner classes should be protected or private	OPT.JAVA.RGS.PKG	PKG: Inner classes should be 'protected' or 'private'
Medium	Avoid EJB AWT Swing	OPT.JAVA.SEC_JAVA.AvoidEJBAWTSwing	AvoidEJBAWTSwing: EJB Bad Practices: Use of AWT Swing
Medium	Avoid EJB JVM Shutdown	OPT.JAVA.SEC_JAVA.AvoidEJBVMShutdown	AvoidEJBVMShutdown: J2EE Bad Practices: Use of System.exit()
Medium	Avoid EJB Java Io	OPT.JAVA.SEC_JAVA.AvoidEJBJavaIo	AvoidEJBJavaIo: EJB Bad Practices: Use of Java I/O
Medium	Avoid EJB Redirect Streams	OPT.JAVA.SEC_JAVA.AvoidEJBRedirectStreams	AvoidEJBRedirectStreams: Avoid changing the input, output, and error streams in EJB

Severity	Contrast rule	Engine rule ID	Description
Medium	Avoid EJB Set Class Loader	OPT.JAVA.SEC_JAVA.AvoidEJBSetClassLoader	AvoidEJBSetClassLoader: Avoid setting context ClassLoader in EJB
Medium	Avoid EJB Set Security Manager	OPT.JAVA.SEC_JAVA.AvoidEJBSetSecurityManager	AvoidEJBSetSecurityManager: Avoid setting system SecurityManager in EJB
Medium	Avoid EJB Synchronization Primitives	OPT.JAVA.SEC_JAVA.AvoidEJBsynchronizationPrimitives	AvoidEJBsynchronizationPrimitives: Avoid use of synchronization primitives in EJB
Medium	Avoid J2EE Direct Database Connection	OPT.JAVA.SEC_JAVA.AvoidJ2EEDirectDatabaseConnection	AvoidJ2EEDirectDatabaseConnection: J2EE Bad Practices: Direct Management of Connections
Medium	Avoid J2EE Explicit Socket	OPT.JAVA.SEC_JAVA.AvoidJ2EEExplicitSocket	AvoidJ2EEExplicitSocket: J2EE Bad Practices: Direct Use of Sockets
Medium	Avoid J2EE Explicit Thread Management	OPT.JAVA.SEC_JAVA.AvoidJ2EEExplicitThreadManagement	AvoidJ2EEExplicitThreadManagement: J2EE Bad Practices: Direct Use of Threads
Medium	Avoid J2EE Leftover Debug Code	OPT.JAVA.SEC_JAVA.AvoidJ2EELeftoverDebugCode	AvoidJ2EELeftoverDebugCode: Leftover Debug Code in J2EE applications
Medium	Avoid J2EE Non Serializable Objects Stored	OPT.JAVA.SEC_JAVA.AvoidJ2EENonSerializableObjectsStored	AvoidJ2EENonSerializableObjectsStored: Avoid non-serializable objects stored in session in J2EE applications
Medium	Avoid Native Calls Rule	OPT.JAVA.SEC_JAVA.AvoidNativeCallsRule	AvoidNativeCallsRule: Avoid calls from Java to native (JNI) code
Medium	Detail Error Leak Rule	OPT.JAVA.SEC_JAVA.DetailErrorLeakRule	DetailErrorLeakRule: Do not send detail error information to client
Medium	Execution After Redirect	OPT.JAVA.SEC_JAVA.ExecutionAfterRedirect	ExecutionAfterRedirect: Execution After Redirect (EAR)
Medium	Potential Infinite Loop	OPT.JAVA.SEC_JAVA.PotentialInfiniteLoop	PotentialInfiniteLoop: Loop with Unreachable Exit Condition ('Infinite Loop')
Medium	Unchecked Input In Loop Condition	OPT.JAVA.SEC_JAVA.UncheckedInputInLoopCondition	UncheckedInputInLoopCondition: Unchecked input in loop condition
Medium	Avoid Assign Repeated Value	OPT.JAVA.SENT.AvoidAssignRepeatedValue	AvoidAssignRepeatedValue: Avoid assigning a value to a variable that has already been assigned
Medium	Avoid Call Get Resource Rule	OPT.JAVA.SENT.AvoidCallGetResourceRule	AvoidCallGetResourceRule: Avoid calling this.getClass.getResource
Medium	Avoid Compare Float Point	OPT.JAVA.SENT.AvoidCompareFloatPoint	AvoidCompareFloatPoint: Avoid comparing floating point with Code Quality[{}]
Medium	Implement Externalizable	OPT.JAVA.SERI.ImplementExternalizable	ImplementExternalizable: Every class that implements Externalizable must have a constructor with no parameters
Medium	Private Read Write Objects	OPT.JAVA.SERI.PrivateReadWriteObjects	PrivateReadWriteObjects: Declare optional readObject and writeObject as private in externalizable / serializable classes
Medium	Avoid Implement Single Thread Model	OPT.JAVA.SERV.AvoidImplementSingleThreadModel	AvoidImplementSingleThreadModel: Avoid implementing the interface javax.servlet.SingleThreadModel in a servlet
Medium	Avoid Large Synchronised Blocks	OPT.JAVA.SERV.AvoidLargeSynchronisedBlocks	AvoidLargeSynchronisedBlocks: Avoid synchronized blocks with more than 6 statements
Medium	Avoid Constructor Injection By Name	OPT.JAVA.SPRING.AvoidConstructorInjectionByName	AvoidConstructorInjectionByName: Avoid constructor injection by name

Severity	Contrast rule	Engine rule ID	Description
Medium	Avoid Hardcoding Properties Into XML	OPT.JAVA.SPRING.AvoidHardcodingPropertiesIntoXML	AvoidHardcodingPropertiesIntoXML: Avoid hardcoding properties into XML descriptors, externalize them instead
Medium	Avoid Loading Multiple XML Configuration Files	OPT.JAVA.SPRING.AvoidLoadingMultipleXMLConfigurationFiles	AvoidLoadingMultipleXMLConfigurationFiles: Avoid loading multiple XML configuration files
Medium	Avoid Too Deep Hierarchy In Jobs	OPT.JAVA.SPRING.AvoidTooDeepHierarchyInJobs	AvoidTooDeepHierarchyInJobs: Avoid too deep jobs hierarchy
Medium	Avoid Too Deep Hierarchy In Steps	OPT.JAVA.SPRING.AvoidTooDeepHierarchyInSteps	AvoidTooDeepHierarchyInSteps: Avoid too deep steps hierarchy
Medium	Avoid Using Version Numbers Within Spring Schema Names	OPT.JAVA.SPRING.AvoidUsingVersionNumbersWithinSpringSchemaNames	AvoidUsingVersionNumbersWithinSpringSchemaNames: Avoid using version numbers in Spring schema names
Medium	Disable XML Validation In Production	OPT.JAVA.SPRING.DisableXMLValidationInProduction	DisableXMLValidationInProduction: Disable XML validation in production
Medium	Enforces JSON Responses	OPT.JAVA.SPRING.EnforcesJSONResponses	EnforcesJSONResponses: Use JSON responses in REST applications
Medium	Prefer Repository Rest Resource Annotation	OPT.JAVA.SPRING.PreferRepositoryRestResourceAnnotation	PreferRepositoryRestResourceAnnotation: Use RepositoryRestResource annotation instead of RestResource
Medium	Setup Commit Interval For Chunk Processing	OPT.JAVA.SPRING.SetupCommitIntervalForChunkProcessing	SetupCommitIntervalForChunkProcessing: Set up a proper commit interval for chunk processing
Medium	Use Annotations To Create Controllers	OPT.JAVA.SPRING.UseAnnotationsToCreateControllers	UseAnnotationsToCreateControllers: Create controllers just using annotations
Medium	Use Java Based Configuration Instead XML Config	OPT.JAVA.SPRING.UseJavaBasedConfigurationInsteadXMLConfig	UseJavaBasedConfigurationInsteadXMLConfig: Use Java based configuration instead of XML based configuration
Medium	Use Lazy Initialized Singleton Beans	OPT.JAVA.SPRING.UseLazyInitializedSingletonBeans	UseLazyInitializedSingletonBeans: Use lazy initialized singleton beans
Medium	Use Thread Safe Multi Threading Steps	OPT.JAVA.SPRING.UseThreadSafeMultiThreadingSteps	UseThreadSafeMultiThreadingSteps: Use thread safe multi threading steps
Medium	A D L R	OPT.JAVA.STR.ADLR	ADLR: Avoid duplicate literals
Medium	Avoid String Concat	OPT.JAVA.STR.AvoidStringConcat	AvoidStringConcat: Avoid String concatenation using + or + {}
Medium	L S T R	OPT.JAVA.STR.LSTR	LSTR: Do not concatenate single character strings in a loop
Medium	N T U L C	OPT.JAVA.STR.NTULC	NTULC: Never use either toUpperCase.equals or toLowerCase.equals
Medium	S B	OPT.JAVA.STR.SB	SB: Instantiate StringBuffer/StringBuilder with explicit initial size
Medium	U C T M	OPT.JAVA.STR.UCTM	UCTM: Avoid unnecessary temporal conversion

Severity	Contrast rule	Engine rule ID	Description
Medium	U S B	OPT.JAVA.STR.USB	USB: Avoid using the concatenation operator of strings + {}
Medium	F I E L D S	OPT.JAVA.STRUTS.FIELDS	FIELDS: Provide a getter and a setter for each field in a form Bean that extends ActionForm
Medium	F O R M	OPT.JAVA.STRUTS.FORM	FORM: Use only getters and setters in form beans
Medium	I N S T	OPT.JAVA.STRUTS.INST	INST: Avoid instance variables in an Action class
Medium	Case With Return	OPT.JAVA.SWITCH.CaseWithReturn	CaseWithReturn: Every CASE statement should end with a break statement
Medium	Avoid Map Set In URL	OPT.JAVA.VELOC.AvoidMapSetInURL	AvoidMapSetInURL: Avoid Map and Set methods of java.net.URL objects
Medium	Avoid String Equals	OPT.JAVA.VELOC.AvoidStringEquals	AvoidStringEquals: Do not check the length of a series by invoking the String.equals("")
Medium	Avoid URL Methods	OPT.JAVA.VELOC.AvoidURLMethods	AvoidURLMethods: Avoid using the equals() and hashCode() in java.net.URL
Medium	Transient For System Resources	OPT.JAVA.J2SE.TransientForSystemResources	TransientForSystemResources: Mark as transient the fields with system resources
Medium	Check HTTP Methods	OPT.JAVA.JAX.CheckHTTPMethods	CheckHTTPMethods: Check the HTTP method used to send the request
Medium	Avoid Total Memory Debug	OPT.JAVA.PERF.AvoidTotalMemoryDebug	AvoidTotalMemoryDebug: Avoid debugging with Runtime.totalMemory()
Medium	Avoid Trace Method Calls Debug	OPT.JAVA.PERF.AvoidTraceMethodCallsDebug	AvoidTraceMethodCallsDebug: Avoid debugging with Runtime.traceMethodCalls ()
Medium	Hardcoded Username Password	OPT.JAVA.SEC_JAVA.HardcodedUsernamePassword	HardcodedUsernamePassword: Use of Hard-coded Credentials
Medium	Password In Configuration File	OPT.JAVA.SEC_JAVA.PasswordInConfigurationFile	PasswordInConfigurationFile: Use of credentials into configuration file
Medium	Plaintext Storage Of Password	OPT.JAVA.SEC_JAVA.PlaintextStorageOfPassword	PlaintextStorageOfPassword: Plaintext Storage of a Password
Medium	Serializable Class Containing Sensitive Data	OPT.JAVA.SEC_JAVA.SerializableClassContainingSensitiveData	SerializableClassContainingSensitiveData: Serializable Class Containing Sensitive Data
Medium	Weak Password Hashing	OPT.JAVA.SEC_JAVA.WeakPasswordHashing	WeakPasswordHashing: Weak Password Hashing

JavaScript Scan rules

Contrast Scan supports these rules for JavaScript.

Severity	Contrast rule	Engine rule ID	Description
Critical	Improper Certificate Validation	OPT.JAVASCRIPT.ImproperCertificateValidation	ImproperCertificateValidation: Improper Certificate Validation
Critical	Too Much Origins Allowed	OPT.JAVASCRIPT.TooMuchOriginsAllowed	TooMuchOriginsAllowed: CORS policy (Cross-origin resource sharing) too permissive
Critical	Contextual Escaping Disabled	OPT.JAVASCRIPT.ANGULARJS.ContextualEscapingDisabled	ContextualEscapingDisabled: Strict Contextual Escaping (SCE) disabled

Severity	Contrast rule	Engine rule ID	Description
Critical	Unsafe Resource Url Whitelist	OPT.JAVASCRIPT.ANGULARJS.UnsafeResourceUrlWhitelist	UnsafeResourceUrlWhitelist: Loading Angular templates insecurely
Critical	Unsafe Url Whitelist	OPT.JAVASCRIPT.ANGULARJS.UnsafeUrlWhitelist	UnsafeUrlWhitelist: Unsafe URL whitelisting
Critical	Sandbox Allow Scripts And Same Origin	OPT.JAVASCRIPT.JSX.SandboxAllowScriptsAndSameOrigin	SandboxAllowScriptsAndSameOrigin: Unsafe sandbox with allow-scripts and same-origin
Critical	No Use Of Eval	OPT.JAVASCRIPT.PERFORMANCE.NoUseOfEval	NoUseOfEval: Do not use eval() for security and performance reasons
Critical	Client Side Template Injection	OPT.JAVASCRIPT.ClientSideTemplateInjection	ClientSideTemplateInjection: Client-side Template Injection
Critical	Code Injection	OPT.JAVASCRIPT.CodeInjection	CodeInjection: Improper Control of Generation of Code ('Code Injection')
Critical	Code Injection With Deserialization	OPT.JAVASCRIPT.CodeInjectionWithDeserialization	CodeInjectionWithDeserialization: Improper code injection during object deserialization
Critical	Command Injection	OPT.JAVASCRIPT.CommandInjection	CommandInjection: Avoid non-neutralized user-controlled input to be part of a command
Critical	Connection String Parameter Pollution	OPT.JAVASCRIPT.ConnectionStringParameterPollution	ConnectionStringParameterPollution: Connection string polluted with untrusted input
Critical	Cookie Poisoning	OPT.JAVASCRIPT.CookiePoisoning	CookiePoisoning: Cookie Poisoning
Critical	Cross Site Scripting	OPT.JAVASCRIPT.CrossSiteScripting	CrossSiteScripting: Improper neutralization of input during web content generation (Cross-site Scripting, XSS)
Critical	DoS Regexp	OPT.JAVASCRIPT.DoSRegexp	DoSRegexp: Potential denial-of-service attack through malicious regular expressions (ReDoS)
Critical	Http Parameter Pollution	OPT.JAVASCRIPT.HttpParameterPollution	HttpParameterPollution: HTTP parameter pollution (HPP)
Critical	Ldap Injection	OPT.JAVASCRIPT.LdapInjection	LdapInjection: Avoid non-neutralized user-controlled input in LDAP search filters
Critical	Mail Command Injection	OPT.JAVASCRIPT.MailCommandInjection	MailCommandInjection: Mail Command Injection
Critical	No SQL Injection	OPT.JAVASCRIPT.NoSQLInjection	NoSQLInjection: Improper neutralization of special elements in data query logic (SQL injection)
Critical	Resource Injection	OPT.JAVASCRIPT.ResourceInjection	ResourceInjection: Do not allow external input to control resource identifiers
Critical	Same Origin Method Execution	OPT.JAVASCRIPT.SameOriginMethodExecution	SameOriginMethodExecution: Same-Origin Method Execution (SOME)
Critical	Server Side Template Injection	OPT.JAVASCRIPT.ServerSideTemplateInjection	ServerSideTemplateInjection: Server-side Template Injection
Critical	SQL Injection	OPT.JAVASCRIPT.SqlInjection	SqlInjection: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
Critical	Stored Cross Site Scripting	OPT.JAVASCRIPT.StoredCrossSiteScripting	StoredCrossSiteScripting: Web content generation from improperly sanitized data and escaped output (Stored Cross Site Scripting, XSS)
Critical	Xml Entity Injection	OPT.JAVASCRIPT.XmlEntityInjection	XmlEntityInjection: XML entity injection
Critical	Angular Cross Site Scripting	OPT.JAVASCRIPT.ANGULARJS.AngularCrossSiteScripting	AngularCrossSiteScripting: Improper neutralization of input during web content generation (Cross-site Scripting, XSS) AngularJS

Severity	Contrast rule	Engine rule ID	Description
Critical	Vue Html Escape Disabled	OPT.JAVASCRIPT.VUE.VueHtmlEscapeDisabled	VueHtmlEscapeDisabled: Vue HTML escaping is disabled.
Critical	Avoid Assignment In Condition	OPT.JAVASCRIPT.ERRORCOMUN.AvoidAssignmentInCondition	AvoidAssignmentInCondition: Avoid assignments into conditional statements
Critical	Avoid Loop With Empty Body	OPT.JAVASCRIPT.ERRORCOMUN.AvoidLoopWithEmptyBody	AvoidLoopWithEmptyBody: Avoid loops (while, do/while, for) with empty bodies
Critical	Avoid Unary Ops In Assign	OPT.JAVASCRIPT.ERRORCOMUN.AvoidUnaryOpsInAssign	AvoidUnaryOpsInAssign: Avoid unary increment or decrement of a variable
Critical	No Update Loop Vars In For Body	OPT.JAVASCRIPT.ERRORCOMUN.NoUpdateLoopVarsInForBody	NoUpdateLoopVarsInForBody: Do not update control vars in 'for' loop bodies
Critical	Avoid Big Files	OPT.JAVASCRIPT.ESTILO.AvoidBigFiles	AvoidBigFiles: Avoid too big JavaScript files
Critical	Avoid Large Functions	OPT.JAVASCRIPT.ESTILO.AvoidLargeFunctions	AvoidLargeFunctions: Avoid functions with excessive number of lines
Critical	Avoid Popup Windows	OPT.JAVASCRIPT.ESTILO.AvoidPopupWindows	AvoidPopupWindows: Avoid popup windows
Critical	Avoid Document All	OPT.JAVASCRIPT.PORTABILITY.AvoidDocumentAll	AvoidDocumentAll: Do not use document.all or document.layers
Critical	Avoid Overwriting Builtin Objects	OPT.JAVASCRIPT.AvoidOverwritingBuiltinObjects	AvoidOverwritingBuiltinObjects: Avoid overwriting JavaScript built-in objects
Critical	Path Manipulation	OPT.JAVASCRIPT.PathManipulation	PathManipulation: External Control Name or Path
Critical	Avoid Cyclic Dependencies	OPT.JAVASCRIPT.NODEJS.AvoidCyclicDependencies	AvoidCyclicDependencies: Avoid cyclic dependencies between modules
Critical	Avoid Using Process Exit	OPT.JAVASCRIPT.NODEJS.AvoidUsingProcessExit	AvoidUsingProcessExit: Avoid using process.exit()
Critical	Avoid Dom Manipulation In Controllers	OPT.JAVASCRIPT.ANGULARJS.AvoidDomManipulationInControllers	AvoidDomManipulationInControllers: DOM manipulation in controllers
Critical	Bind Objects In Scope	OPT.JAVASCRIPT.ANGULARJS.BindObjectsInScope	BindObjectsInScope: Bind to object scope, instead of binding to property
Critical	Deprecated Directive Format	OPT.JAVASCRIPT.ANGULARJS.DeprecatedDirectiveFormat	DeprecatedDirectiveFormat: Avoid deprecated directive formats
Critical	Never Store Dom In Scope	OPT.JAVASCRIPT.ANGULARJS.NeverStoreDomInScope	NeverStoreDomInScope: Never store elements in scope
Critical	Private Property Access	OPT.JAVASCRIPT.ANGULARJS.PrivatePropertyAccess	PrivatePropertyAccess: Do not access private properties of AngularJS objects
Critical	Unsafe Minification Annotation	OPT.JAVASCRIPT.ANGULARJS.UnsafeMinificationAnnotation	UnsafeMinificationAnnotation: Use minification-safe annotations in dependency injection
Critical	Use Controller As Syntax In Views	OPT.JAVASCRIPT.ANGULARJS.UseControllerAsSyntaxInViews	UseControllerAsSyntaxInViews: Use "controller as" syntax in views
Critical	Watch Collection Change	OPT.JAVASCRIPT.ANGULARJS.WatchCollectionChange	WatchCollectionChange: Use \$watchCollection instead of \$watch with three parameters
Critical	Too Broad Access Origin	OPT.JAVASCRIPT.CORDOVA.TooBroadAccessOrigin	TooBroadAccessOrigin: Access policy too broad
Critical	Vue Component Data Must Be Function	OPT.JAVASCRIPT.VUE.VueComponentDataMustBeFunction	VueComponentDataMustBeFunction: Component data must be a function
Critical	Missing Password Field Masking	OPT.JAVASCRIPT.JSX.MissingPasswordFieldMasking	MissingPasswordFieldMasking: Password input field is not masked

Severity	Contrast rule	Engine rule ID	Description
High	Clickjacking Protection	OPT.JAVASCRIPT.ClickjackingProtection	ClickjackingProtection: No clickjack protection configured
High	Plaintext Storage In A Cookie	OPT.JAVASCRIPT.PlaintextStorageInACookie	PlaintextStorageInACookie: Cleartext Storage of Sensitive Information in Cookie
High	Use Strict Transport Security	OPT.JAVASCRIPT.UseStrictTransportSecurity	UseStrictTransportSecurity: Use HTTP Strict Transport Security
High	Xss Protection Disabled	OPT.JAVASCRIPT.XssProtectionDisabled	XssProtectionDisabled: Cross-site scripting protection disabled
High	Avoid Enabled Debug Mode	OPT.JAVASCRIPT.CORDOVA.AvoidEnabledDebugMode	AvoidEnabledDebugMode: Debug mode enabled
High	Insecure Android Min Sdk Version	OPT.JAVASCRIPT.CORDOVA.InsecureAndroidMinSdkVersion	InsecureAndroidMinSdkVersion: Android SDK version too old
High	Whitelist Plugin Not Installed	OPT.JAVASCRIPT.CORDOVA.WhitelistPluginNotInstalled	WhitelistPluginNotInstalled: Whitelist plugin not installed
High	Cross Site Request Forgery	OPT.JAVASCRIPT.CrossSiteRequestForgery	CrossSiteRequestForgery: Execution of action on user behalf in a previously authenticated web site (cross-site request forgery, CSRF)
High	External Control Of Configuration Setting	OPT.JAVASCRIPT.ExternalControlOfConfigurationSetting	ExternalControlOfConfigurationSetting: External Control of System or Configuration Setting
High	Header Manipulation	OPT.JAVASCRIPT.HeaderManipulation	HeaderManipulation: Unvalidated data in HTTP response header or in cookie (Response Splitting)
High	Open Redirect	OPT.JAVASCRIPT.OpenRedirect	OpenRedirect: URL Redirection to another Site ('Open Redirect')
High	Open Redirect Hana XS	OPT.JAVASCRIPT.OpenRedirectHanaXS	OpenRedirectHanaXS: Open Redirect (HANA XS)
High	Server Side Request Forgery	OPT.JAVASCRIPT.ServerSideRequestForgery	ServerSideRequestForgery: Creating requests from a vulnerable server using untrusted input (server side request forgery, SSRF)
High	XPath Injection	OPT.JAVASCRIPT.XPathInjection	XPathInjection: Improper Neutralization of Data within XPath Expressions ('XPath Injection')
High	Target Blank Vulnerability	OPT.JAVASCRIPT.JSX.TargetBlankVulnerability	TargetBlankVulnerability: Improper Neutralization of links to external sites
High	Avoid Empty Functions	OPT.JAVASCRIPT.ERRORCOMUN.AvoidEmptyFunctions	AvoidEmptyFunctions: Avoid top-level functions with empty body
High	Many Cases	OPT.JAVASCRIPT.ERRORCOMUN.ManyCases	ManyCases: Avoid too many choices in switch structures
High	Potential Infinite Loop	OPT.JAVASCRIPT.ERRORCOMUN.PotentialInfiniteLoop	PotentialInfiniteLoop: Potential infinite loop
High	Unused Function Parameter	OPT.JAVASCRIPT.ERRORCOMUN.UnusedFunctionParameter	UnusedFunctionParameter: Avoid unused function parameters
High	Unused Local Var	OPT.JAVASCRIPT.ERRORCOMUN.UnusedLocalVar	UnusedLocalVar: Avoid unused local variable
High	Avoid Conditional Operator	OPT.JAVASCRIPT.ESTILO.AvoidConditionalOperator	AvoidConditionalOperator: Do not use ternary operator to evaluate conditions
High	Avoid Declaring Vars Without Var	OPT.JAVASCRIPT.ESTILO.AvoidDeclaringVarsWithoutVar	AvoidDeclaringVarsWithoutVar: Declare variables with var
High	Avoid Using With	OPT.JAVASCRIPT.ESTILO.AvoidUsingWith	AvoidUsingWith: Avoid using 'with' statement

Severity	Contrast rule	Engine rule ID	Description
High	End Sentences With Semicolon	OPT.JAVASCRIPT.ESTILO.EndSentencesWithSemicolon	EndSentencesWithSemicolon: Avoid statements without semicolon
High	Avoid Non Portable Methods	OPT.JAVASCRIPT.PORTABILITY.AvoidNonPortableMethods	AvoidNonPortableMethods: Non-portable function check
High	No Navigator For Browser Detection	OPT.JAVASCRIPT.PORTABILITY.NoNavigatorForBrowserDetection	NoNavigatorForBrowserDetection: Avoid using navigator.userAgent ('browser detecting') for writing portable code
High	Avoid Accessing Unreliable Variable Properties	OPT.JAVASCRIPT.AvoidAccessingUnreliableVariableProperties	AvoidAccessingUnreliableVariableProperties: Avoid accessing unreliable variable properties
High	Avoid Calling Too Many Other Components	OPT.JAVASCRIPT.AvoidCallingTooManyOtherComponents	AvoidCallingTooManyOtherComponents: Avoid using components calling too many other components
High	Avoid Misuse Of Delete	OPT.JAVASCRIPT.AvoidMisuseOfDelete	AvoidMisuseOfDelete: Delete operator should be only properly used with object properties
High	Avoid Named Functions	OPT.JAVASCRIPT.AvoidNamedFunctions	AvoidNamedFunctions: Avoid defining functions in conditional blocks
High	Avoid Object Instantiation Into Loops	OPT.JAVASCRIPT.AvoidObjectInstantiationIntoLoops	AvoidObjectInstantiationIntoLoops: Avoid object instantiation into loops
High	Avoid Too Complex Functions	OPT.JAVASCRIPT.AvoidTooComplexFunctions	AvoidTooComplexFunctions: Avoid methods with high cyclomatic complexity values
High	Avoid Too Complex Programs	OPT.JAVASCRIPT.AvoidTooComplexPrograms	AvoidTooComplexPrograms: Avoid classes with high cyclomatic complexity values
High	Avoid Using Unary Operators With Objects	OPT.JAVASCRIPT.AvoidUsingUnaryOperatorsWithObjects	AvoidUsingUnaryOperatorsWithObjects: Avoid using the + and - unary operators on objects
High	Duplicated Name For Function And Variable	OPT.JAVASCRIPT.DuplicatedNameForFunctionAndVariable	DuplicatedNameForFunctionAndVariable: Avoid declaring a function with the same name as a variable
High	Function Arguments Uniqueness	OPT.JAVASCRIPT.FunctionArgumentsUniqueness	FunctionArgumentsUniqueness: Avoid duplicated argument names in function declarations
High	IE Conditional Comments	OPT.JAVASCRIPT.IEConditionalComments	IEConditionalComments: Avoid using Internet Explorer conditional comments
High	Nested If Statements	OPT.JAVASCRIPT.NestedIfStatements	NestedIfStatements: Avoid a high number of nested ifs
High	Property Names Uniqueness	OPT.JAVASCRIPT.PropertyNamesUniqueness	PropertyNamesUniqueness: Avoid duplicating property names in objects
High	Unhandled Promise	OPT.JAVASCRIPT.UnhandledPromise	UnhandledPromise: Handle function returned promises
High	Variable Redclaration	OPT.JAVASCRIPT.VariableRedeclaration	VariableRedeclaration: Avoid declaring variable with a name that is already declared
High	Avoid Too Much Nested Callbacks	OPT.JAVASCRIPT.NODEJS.AvoidTooMuchNestedCallbacks	AvoidTooMuchNestedCallbacks: Avoid too many nested callbacks
High	Avoid Using Default Connection Limit	OPT.JAVASCRIPT.NODEJS.AvoidUsingDefaultConnectionLimit	AvoidUsingDefaultConnectionLimit: Avoid using the default connections limit
High	Validate Package Json	OPT.JAVASCRIPT.NODEJS.ValidatePackageJson	ValidatePackageJson: Avoid specifying dependencies versions with the * wildcard

Severity	Contrast rule	Engine rule ID	Description
High	Require Modules At The Begin	OPT.JAVASCRIPT.RequireModulesAtTheBegin	RequireModulesAtTheBegin: Always require modules at the top of the file
High	Avoid Complex Expressions In Html	OPT.JAVASCRIPT.ANGULARJS.AvoidComplexExpressionsInHtml	AvoidComplexExpressionsInHtml: Avoid complex AngularJS expressions in HTML
High	Avoid Root Scope Event Listeners In Controllers	OPT.JAVASCRIPT.ANGULARJS.AvoidRootScopeEventListenersInControllers	AvoidRootScopeEventListenersInC: Avoid registering event listeners on \$rootScope in controllers
High	Deprecated Http Functions	OPT.JAVASCRIPT.ANGULARJS.DeprecatedHttpFunctions	DeprecatedHttpFunctions: Do not use deprecated \$http functions
High	Ng Src When Using Expressions	OPT.JAVASCRIPT.ANGULARJS.NgSrcWhenUsingExpressions	NgSrcWhenUsingExpressions: Always use ng-src for images when including an AngularJS expression
High	Prevent Component Name Collision	OPT.JAVASCRIPT.ANGULARJS.PreventComponentNameCollision	PreventComponentNameCollision: Prevent name collision in AngularJS component definition
High	Resolve Controller Dependencies In Route	OPT.JAVASCRIPT.ANGULARJS.ResolveControllerDependenciesInRoute	ResolveControllerDependenciesInF: Resolve controller dependencies in route
High	Restrict Directives Element Attribute	OPT.JAVASCRIPT.ANGULARJS.RestrictDirectivesElementAttribute	RestrictDirectivesElementAttribute: Restrict directives to elements and attributes
High	Use Named Functions For Components	OPT.JAVASCRIPT.ANGULARJS.UseNamedFunctionsForComponents	UseNamedFunctionsForComponent: Use named functions instead of callback functions for components
High	Avoid Annotating Inferable Types	OPT.JAVASCRIPT.TYPESCRIPT.AvoidAnnotatingInferableTypes	AvoidAnnotatingInferableTypes: Avoid type annotations for inferable primitive types
High	No Empty Interface	OPT.JAVASCRIPT.TYPESCRIPT.NoEmptyInterface	NoEmptyInterface: Avoid using empty interfaces
High	Prefer Read Only	OPT.JAVASCRIPT.TYPESCRIPT.PreferReadOnly	PreferReadOnly: Use readonly when a property is never reassigned
High	Skip Internal Module Or Namespace	OPT.JAVASCRIPT.TYPESCRIPT.SkipInternalModuleOrNamespace	SkipInternalModuleOrNamespace: Skip ES2015 module syntax
High	Useless Type Cast	OPT.JAVASCRIPT.TYPESCRIPT.UselessTypeCast	UselessTypeCast: Avoid useless type castings
High	Useless Type Intersection	OPT.JAVASCRIPT.TYPESCRIPT.UselessTypeIntersection	UselessTypeIntersection: Avoid useless intersection
High	Use Type Annotations	OPT.JAVASCRIPT.TYPESCRIPT.UseTypeAnnotations	UseTypeAnnotations: Use TypeScript type system
High	Avoid Forward Refs	OPT.JAVASCRIPT.TYPESCRIPT.ANGULAR.AvoidForwardRefs	AvoidForwardRefs: Avoid using the forwardRef function
High	Avoid Impure Pipes	OPT.JAVASCRIPT.TYPESCRIPT.ANGULAR.AvoidImpurePipes	AvoidImpurePipes: Avoid impure Pipes
High	Avoid Template Async Negation	OPT.JAVASCRIPT.TYPESCRIPT.ANGULAR.AvoidTemplateAsyncNegation	AvoidTemplateAsyncNegation: Inconsistent Async Pipe usage in templates.
High	Decorator Incompatibility	OPT.JAVASCRIPT.TYPESCRIPT.ANGULAR.DecoratorIncompatibility	DecoratorIncompatibility: Avoid using decorators with incompatibilities between them
High	Use Host Decorator	OPT.JAVASCRIPT.TYPESCRIPT.ANGULAR.UseHostDecorator	UseHostDecorator: Use @Host decorator instead of host metadata property
High	Use Injectable Decorator	OPT.JAVASCRIPT.TYPESCRIPT.ANGULAR.UseInjectableDecorator	UseInjectableDecorator: Use @Injectable class decorator instead of the @Inject parameter decorator

Severity	Contrast rule	Engine rule ID	Description
High	Use Input Decorator	OPT.JAVASCRIPT.TYPESCRIPT.ANGULAR.UseInputDecorator	UseInputDecorator: Use @Input decorator instead of inputs metadata property
High	Use Output Decorator	OPT.JAVASCRIPT.TYPESCRIPT.ANGULAR.UseOutputDecorator	UseOutputDecorator: Use @Output decorator instead of inputs metadata property
High	Use Track By	OPT.JAVASCRIPT.TYPESCRIPT.ANGULAR.UseTrackBy	UseTrackBy: Use trackBy along with
High	Avoid Click Events	OPT.JAVASCRIPT.CORDOVA.AvoidClickEvents	AvoidClickEvents: Avoid using click events in Cordova.
High	Vue For Without Key	OPT.JAVASCRIPT.VUE.VueForWithoutKey	VueForWithoutKey: Always use key with v-for.
High	Vue If With For Directive	OPT.JAVASCRIPT.VUE.VuelfWithForDirective	VuelfWithForDirective: Never use v-if with the same element as v-for.
High	Avoid Web SQL	OPT.JAVASCRIPT.AvoidWebSQL	AvoidWebSQL: Avoid Web SQL
High	Empty Or Hardcoded Password	OPT.JAVASCRIPT.EmptyOrHardcodedPassword	EmptyOrHardcodedPassword: Empty or hardcoded passwords may compromise system security in a way that cannot be easily remedied
High	Prevent MIME Sniffing	OPT.JAVASCRIPT.PreventMIMESniffing	PreventMIMESniffing: Prevent MIME sniffing
High	Angular Local Storage Information Leak	OPT.JAVASCRIPT.ANGULARJS.AngularLocalStorageInformationLeak	AngularLocalStorageInformationLeak: AngularJS local storage information leak
High	Hardcoded Crypto Key	OPT.JAVASCRIPT.HardcodedCryptoKey	HardcodedCryptoKey: Hardcoded cryptographic keys
High	Insecure Transport	OPT.JAVASCRIPT.InsecureTransport	InsecureTransport: Insecure transport
High	Insufficient Key Size	OPT.JAVASCRIPT.InsufficientKeySize	InsufficientKeySize: An otherwise strong encryption algorithm is vulnerable to brute force attack when a small key size is used
High	Server Insecure Transport	OPT.JAVASCRIPT.ServerInsecureTransport	ServerInsecureTransport: Insecure transport in Node.js HTTP servers
High	Weak Cryptographic Hash	OPT.JAVASCRIPT.WeakCryptographicHash	WeakCryptographicHash: Weak cryptographic hash
High	Weak Encryption	OPT.JAVASCRIPT.WeakEncryption	WeakEncryption: Weak symmetric encryption algorithm
Info	Code Document Percentage	OPT.JAVASCRIPT.DOCUMENTACION.CodeDocumentPercentage	CodeDocumentPercentage: Document code percentage
Info	Document Every Function	OPT.JAVASCRIPT.DOCUMENTACION.DocumentEveryFunction	DocumentEveryFunction: Insert header comments before every top-level function
Info	Function Redecclaration	OPT.JAVASCRIPT.FunctionRedeclaration	FunctionRedeclaration: Avoid duplicating function names in same scope
Info	Multiline String Literals	OPT.JAVASCRIPT.MultilineStringLiterals	MultilineStringLiterals: Avoid splitting string literal in multiple lines using \n character
Info	Avoid Using Process Env	OPT.JAVASCRIPT.NODEJS.AvoidUsingProcessEnv	AvoidUsingProcessEnv: Avoid using process.env()
Info	Module Definition And Use	OPT.JAVASCRIPT.ANGULARJS.ModuleDefinitionAndUse	ModuleDefinitionAndUse: Declare and access modules using setter/getter without creating a variable
Low	Form Without Captcha	OPT.JAVASCRIPT.JSX.FormWithoutCaptcha	FormWithoutCaptcha: Form without CAPTCHA
Low	Use Space Between Operators	OPT.JAVASCRIPT.ESTILO.UseSpaceBetweenOperators	UseSpaceBetweenOperators: Place whitespaces between logical operators and its operands
Low	Global Var Pattern	OPT.JAVASCRIPT.JSNOM.GlobalVarPattern	GlobalVarPattern: Global vars should be avoided or must follow a naming pattern

Severity	Contrast rule	Engine rule ID	Description
Low	Identifier Naming Pattern	OPT.JAVASCRIPT.JSNOM.IdentifierNamingPattern	IdentifierNamingPattern: Follow naming standards for JavaScript identifiers
Low	Avoid Arguments	OPT.JAVASCRIPT.AvoidArguments	AvoidArguments: Do not use arguments object
Low	Avoid Array And Object Constructors	OPT.JAVASCRIPT.AvoidArrayAndObjectConstructors	AvoidArrayAndObjectConstructors: Avoid using Array and Object constructors
Low	Avoid Commented Out Code Blocks	OPT.JAVASCRIPT.AvoidCommentedOutCodeBlocks	AvoidCommentedOutCodeBlocks: Avoid commented out code blocks
Low	Avoid Constructors For Side Effects	OPT.JAVASCRIPT.AvoidConstructorsForSideEffects	AvoidConstructorsForSideEffects: Avoid calling constructors without using its return value
Low	Avoid Function Definition Inside Loop	OPT.JAVASCRIPT.AvoidFunctionDefinitionInsideLoop	AvoidFunctionDefinitionInsideLoop: Avoid declare functions inside loops
Low	Avoid Octal Number	OPT.JAVASCRIPT.AvoidOctalNumber	AvoidOctalNumber: Avoid using octal numbers
Low	Avoid Returning Values From Setters	OPT.JAVASCRIPT.AvoidReturningValuesFromSetters	AvoidReturningValuesFromSetters: Avoid returning a value from setters
Low	Avoid Using Continue	OPT.JAVASCRIPT.AvoidUsingContinue	AvoidUsingContinue: Avoid using 'continue' statement
Low	Avoid Using Debugger	OPT.JAVASCRIPT.AvoidUsingDebugger	AvoidUsingDebugger: Avoid using 'debugger' statement
Low	Break Non Empty Switch Clauses	OPT.JAVASCRIPT.BreakNonEmptySwitchClauses	BreakNonEmptySwitchClauses: Use 'break' statement at the last statement of SwitchCase
Low	Default Clause Switch Statements	OPT.JAVASCRIPT.DefaultClauseSwitchStatements	DefaultClauseSwitchStatements: Use 'default' clause at the end of the switch statements
Low	Else In Else If Statement	OPT.JAVASCRIPT.ElseInElseIfStatement	ElseInElseIfStatement: Else if statements should finish with an else clause
Low	Filter For In	OPT.JAVASCRIPT.FilterForIn	FilterForIn: Filter the body of a for-in statement
Low	Function Declarations Within Blocks	OPT.JAVASCRIPT.FunctionDeclarationsWithinBlocks	FunctionDeclarationsWithinBlocks: Avoid use function declarations within blocks
Low	Labeled Statements	OPT.JAVASCRIPT.LabeledStatements	LabeledStatements: Use labels only for while and do-while statements
Low	One Statement Per Line	OPT.JAVASCRIPT.OneStatementPerLine	OneStatementPerLine: Use only one statement per line
Low	Parent Class Doesnot Reference Child Classes	OPT.JAVASCRIPT.ParentClassDoesnotReferenceChildClasses	ParentClassDoesnotReferenceChildClasses: Parent class does not reference any child classes
Low	Short Circuit If Statements	OPT.JAVASCRIPT.ShortCircuitIfStatements	ShortCircuitIfStatements: Merge nested if statements using a short-circuit operator
Low	Too Many Break Or Continue In Loop	OPT.JAVASCRIPT.TooManyBreakOrContinueInLoop	TooManyBreakOrContinueInLoop: Avoid using more than one break or continue statement in each loop
Low	Trailing Comma	OPT.JAVASCRIPT.TrailingComma	TrailingComma: Avoid using a trailing comma at the end of the last element in the declaration of an array or object
Low	Type Casting In Comparisons	OPT.JAVASCRIPT.TypeCastingInComparisons	TypeCastingInComparisons: Avoid using type casting in logical comparators Code Quality [C]

Severity	Contrast rule	Engine rule ID	Description
Low	Unreachable Code	OPT.JAVASCRIPT.UnreachableCode	UnreachableCode: Return, break, continue, or throw statements should be followed by a } or case or default statements
Low	Use Single Quote	OPT.JAVASCRIPT.UseSingleQuote	UseSingleQuote: Avoid using single quotes in literals
Low	Avoid Mixing Require	OPT.JAVASCRIPT.NODEJS.AvoidMixingRequire	AvoidMixingRequire: Avoid mixing require calls with variable initializations
Low	Use Asynchronous Methods	OPT.JAVASCRIPT.UseAsynchronousMethods	UseAsynchronousMethods: Asynchronous methods give Node.js speed and responsiveness
Low	Use J S Doc	OPT.JAVASCRIPT.UseJSDoc	UseJSDoc: Describe how the function works using JSDoc
Low	Use Module Exports	OPT.JAVASCRIPT.UseModuleExports	UseModuleExports: Use module.exports instead of exports
Low	Isolate Run Blocks	OPT.JAVASCRIPT.ANGULARJS.IsolateRunBlocks	IsolateRunBlocks: Isolate run blocks
Low	Avoid None View Encapsulation	OPT.JAVASCRIPT.TYPESCRIPT.ANGULAR.AvoidNoneViewEncapsulation	AvoidNoneViewEncapsulation: Avoid applying component styles to the whole application
Low	Avoid Prefixing Output	OPT.JAVASCRIPT.TYPESCRIPT.ANGULAR.AvoidPrefixingOutput	AvoidPrefixingOutput: Avoid prefixing properties with "on"
Low	Never Use History	OPT.JAVASCRIPT.ESTILO.NeverUseHistory	NeverUseHistory: Never use JavaScript 'history' object or navigation-based positioning functions
Low	Hide Powered By Header	OPT.JAVASCRIPT.HidePoweredByHeader	HidePoweredByHeader: Deactivate Powered-By header
Low	Password In Comments	OPT.JAVASCRIPT.PasswordInComments	PasswordInComments: Avoid hard-coded in-comment passwords in code
Low	Avoid Using Console For Debugging	OPT.JAVASCRIPT.NODEJS.AvoidUsingConsoleForDebugging	AvoidUsingConsoleForDebugging: Avoid using console.log()
Medium	Unsafe Cookie	OPT.JAVASCRIPT.UnsafeCookie	UnsafeCookie: Generate server-side cookies with adequate security properties
Medium	Avoid Overly Permissive Message Posting	OPT.JAVASCRIPT.AvoidOverlyPermissiveMessagePosting	AvoidOverlyPermissiveMessagePosting: Avoid post cross-document messages to an overly permissive target origin
Medium	Trust Boundary Violation	OPT.JAVASCRIPT.TrustBoundaryViolation	TrustBoundaryViolation: Trust boundary violation
Medium	Specify Integrity Attribute	OPT.JAVASCRIPT.JSX.SpecifyIntegrityAttribute	SpecifyIntegrityAttribute: Specify a integrity attribute on the <script> and <link>
Medium	Javascript Url	OPT.JAVASCRIPT.REACT.JavascriptUrl	JavascriptUrl: Usage of javascript: URLs in JSX.
Medium	Avoid For With External Control Vars	OPT.JAVASCRIPT.ERRORCOMUN.AvoidForWithExternalControlVars	AvoidForWithExternalControlVars: Avoid for loops where loop control vars are not declared in its initialization block
Medium	If Without Block	OPT.JAVASCRIPT.ERRORCOMUN.IfWithoutBlock	IfWithoutBlock: Place body of if statement between braces
Medium	Illegal Identifier	OPT.JAVASCRIPT.ERRORCOMUN.IllegalIdentifier	IllegalIdentifier: Avoid using identifiers not permitted (like reserved keywords)
Medium	Avoid Alert With Literals	OPT.JAVASCRIPT.ESTILO.AvoidAlertWithLiterals	AvoidAlertWithLiterals: Do not use string literals
Medium	Avoid Multiple Returns	OPT.JAVASCRIPT.ESTILO.AvoidMultipleReturns	AvoidMultipleReturns: Avoid functions with more than one return statement
Medium	Check Parameters Number In Function	OPT.JAVASCRIPT.ESTILO.CheckParametersNumberInFunction	CheckParametersNumberInFunction: Avoid using functions with too many parameters
Medium	No Style	OPT.JAVASCRIPT.ESTILO.NoStyle	NoStyle: Do not use style property, use CSS classes instead

Severity	Contrast rule	Engine rule ID	Description
Medium	Avoid Long Calls In Iterations	OPT.JAVASCRIPT.PERFORMANCE.AvoidLongCallsInIterations	AvoidLongCallsInIterations: Avoid long reference chains in loops
Medium	No Method Append Child	OPT.JAVASCRIPT.PERFORMANCE.NoMethodAppendChild	NoMethodAppendChild: Use innerHTML instead of DOM modification functions
Medium	Old Use Of Document	OPT.JAVASCRIPT.PORTABILITY.OldUseOfDocument	OldUseOfDocument: Avoid using non-compliant methods/properties of 'document' object
Medium	Avoid Assigning Undefined	OPT.JAVASCRIPT.AvoidAssigningUndefined	AvoidAssigningUndefined: Avoid assigning undefined to a variable
Medium	Avoid Comparing With Na N	OPT.JAVASCRIPT.AvoidComparingWithNaN	AvoidComparingWithNaN: Avoid comparing with NaN in conditional expressions
Medium	Avoid Magic Numbers	OPT.JAVASCRIPT.AvoidMagicNumbers	AvoidMagicNumbers: Avoid using magic literals
Medium	Avoid Multiple Statements Per Line	OPT.JAVASCRIPT.AvoidMultipleStatementsPerLine	AvoidMultipleStatementsPerLine: Avoid specifying several statements into the same line
Medium	Avoid Negative Content Length	OPT.JAVASCRIPT.AvoidNegativeContentLength	AvoidNegativeContentLength: The Content Length header should not have a negative value
Medium	Avoid Rebinding A Const Variable	OPT.JAVASCRIPT.AvoidRebindingAConstVariable	AvoidRebindingAConstVariable: Avoid rebinding a const variable
Medium	Avoid Too Deep Class Hierarchies	OPT.JAVASCRIPT.AvoidTooDeepClassHierarchies	AvoidTooDeepClassHierarchies: Avoid too deep hierarchy classes
Medium	Avoid Using Parse Int Without Radix	OPT.JAVASCRIPT.AvoidUsingParseIntWithoutRadix	AvoidUsingParseIntWithoutRadix: Always specify a radix when using parseInt
Medium	Denial Of Service	OPT.JAVASCRIPT.DenialOfService	DenialOfService: An attacker could exploit the program becomes unavailable to legitimate users
Medium	Loop Without Block	OPT.JAVASCRIPT.LoopWithoutBlock	LoopWithoutBlock: Place loop body statements between braces
Medium	Avoid Concatenating Dirname And Filename	OPT.JAVASCRIPT.NODEJS.AvoidConcatenatingDirnameAndFilename	AvoidConcatenatingDirnameAndFilename: Avoid concatenating __dirname and __filename with other strings
Medium	Avoid Using New Require	OPT.JAVASCRIPT.NODEJS.AvoidUsingNewRequire	AvoidUsingNewRequire: Avoid invoking a module constructor when importing a module
Medium	Callbacks Always Pass Error Parameter First	OPT.JAVASCRIPT.NODEJS.CallbacksAlwaysPassErrorParameterFirst	CallbacksAlwaysPassErrorParameterFirst: The first callback parameter must be an error
Medium	Ensure Callbacks Are Returned	OPT.JAVASCRIPT.NODEJS.EnsureCallbacksAreReturned	EnsureCallbacksAreReturned: Use return statement along with callback
Medium	Use Gzip Compression	OPT.JAVASCRIPT.NODEJS.UseGzipCompression	UseGzipCompression: Use GZIP compression when using express framework
Medium	Always Use Strict	OPT.JAVASCRIPT.AlwaysUseStrict	AlwaysUseStrict: "use strict" prevents bad practices
Medium	Save A Reference To This	OPT.JAVASCRIPT.SaveAReferenceToThis	SaveAReferenceToThis: The "this" is determined based on context, not encapsulation
Medium	Validate Callbacks	OPT.JAVASCRIPT.ValidateCallbacks	ValidateCallbacks: Only functions are callable
Medium	Define One Component Per File	OPT.JAVASCRIPT.ANGULARJS.DefineOneComponentPerFile	DefineOneComponentPerFile: Define one AngularJS component per file

Severity	Contrast rule	Engine rule ID	Description
Medium	Handle Route Errors	OPT.JAVASCRIPT.ANGULARJS.HandleRouteErrors	HandleRouteErrors: Handle all route errors on a centralised basis
Medium	Use Angular Wrappers	OPT.JAVASCRIPT.ANGULARJS.UseAngularWrappers	UseAngularWrappers: Use Angular wrappers for common objects and functions
Medium	Avoid Casting I Object Literals	OPT.JAVASCRIPT.TYPESCRIPT.AvoidCastingIObjectLiterals	AvoidCastingIObjectLiterals: Avoid casting object literals
Medium	No Return Type Any	OPT.JAVASCRIPT.TYPESCRIPT.NoReturnTypeAny	NoReturnTypeAny: Don't use "any" for function return type
Medium	Review Non Null Assertions	OPT.JAVASCRIPT.TYPESCRIPT.ReviewNonNullAssertions	ReviewNonNullAssertions: Review non-null assertions
Medium	Too Many Classes Per File	OPT.JAVASCRIPT.TYPESCRIPT.TooManyClassesPerFile	TooManyClassesPerFile: Avoid an excessive number of classes per file
Medium	Use Primitive Types	OPT.JAVASCRIPT.TYPESCRIPT.UsePrimitiveTypes	UsePrimitiveTypes: Don't wrap primitive types
Medium	Use Type Alias	OPT.JAVASCRIPT.TYPESCRIPT.UseTypeAlias	UseTypeAlias: Use a type alias when the type is complex
Medium	Avoid Aliasing Input Output	OPT.JAVASCRIPT.TYPESCRIPT.ANGULAR.AvoidAliasingInputOutput	AvoidAliasingInputOutput: Avoid decorating aliases for Input and Output decorators
Medium	Invalid Pipe Implementation	OPT.JAVASCRIPT.TYPESCRIPT.ANGULAR.InvalidPipeImplementation	InvalidPipeImplementation: Implement Angular Pipes completely
Medium	Naming Conventions	OPT.JAVASCRIPT.TYPESCRIPT.ANGULAR.NamingConventions	NamingConventions: Follow naming conventions standards for Angular
Medium	No Parameter Attribute Decorator	OPT.JAVASCRIPT.TYPESCRIPT.ANGULAR.NoParameterAttributeDecorator	NoParameterAttributeDecorator: Avoid decorating constructor parameters with Attribute
Medium	Use Life Cycle Interface	OPT.JAVASCRIPT.TYPESCRIPT.ANGULAR.UseLifeCycleInterface	UseLifeCycleInterface: Use the LifeCycle hook interfaces
Medium	Dangerously Set Inner Html	OPT.JAVASCRIPT.REACT.DangerouslySetInnerHTML	DangerouslySetInnerHTML: Do not use dangerouslySetInnerHTML property on components.
Medium	Find Dom Node	OPT.JAVASCRIPT.REACT.FindDOMNode	FindDOMNode: Do not call ReactDOM.findDOMNode().
Medium	Avoid Transfer Values Local Session Storage	OPT.JAVASCRIPT.AvoidTransferValuesLocalStorage	AvoidTransferValuesLocalStorage: Avoid transferring data between local storage and sessionStorage as it can expose confidential information
Medium	Easy To Guess Database Name	OPT.JAVASCRIPT.EasyToGuessDatabaseName	EasyToGuessDatabaseName: Do not use easy-to-guess Web SQL database names
Medium	Hijacking Ad Hoc Ajax	OPT.JAVASCRIPT.HijackingAdHocAjax	HijackingAdHocAjax: Do not use jQuery to transport sensitive data
Medium	Information Exposure Through Error Message	OPT.JAVASCRIPT.InformationExposureThroughErrorMessage	InformationExposureThroughErrorMessage: Avoid sensitive information exposure through error messages
Medium	Privacy Violation	OPT.JAVASCRIPT.PrivacyViolation	PrivacyViolation: Exposure of Private Information ('Privacy Violation')
Medium	Sensitive Info In Configuration File	OPT.JAVASCRIPT.SensitiveInfoInConfigurationFile	SensitiveInfoInConfigurationFile: Use sensitive information into configuration files
Medium	Autocomplete On For Sensitive Fields	OPT.JAVASCRIPT.JSX.AutocompleteOnForSensitiveFields	AutocompleteOnForSensitiveFields: Autocomplete enabled for sensitive fields
Medium	Insecure Randomness	OPT.JAVASCRIPT.InsecureRandomness	InsecureRandomness: Standard pseudo-random number generators cannot resist cryptographic attacks

JCL Scan rules

Contrast Scan supports these rules for JCL.

Severity	Contrast Rule	Engine rule ID	Description
Critical	Avoid Using Jobs Without Steps	OPT.JCL.AvoidUsingJobsWithoutSteps	AvoidUsingJobsWithoutSteps: Avoid using JCL programs not defining any step
Critical	Job Name Must Match JCL Name	OPT.JCL.NAM_JCL.JobNameMustMatchJCLName	JobNameMustMatchJCLName: JOB name must be equal to the JCL filename
Critical	Condition Code In Steps	OPT.JCL.PB_JCL.ConditionCodeInSteps	ConditionCodeInSteps: Use COND on each step to avoid wasted CPU in case of abnormal end
Critical	Allocate File Space In Tracks	OPT.JCL.PF_JCL.AllocateFileSpaceInTracks	AllocateFileSpaceInTracks: The allocation of space for files must be on tracks (TRKS)
Critical	Max Space For File	OPT.JCL.PF_JCL.MaxSpaceForFile	MaxSpaceForFile: Maximum allocation of space for files
Critical	Time Per Step And Job	OPT.JCL.PF_JCL.TimePerStepAndJob	TimePerStepAndJob: Avoid unlimited time per job or step
High	Avoid Duplicated Step Names	OPT.JCL.AvoidDuplicatedStepNames	AvoidDuplicatedStepNames: Avoid duplicating step names
High	No Steps Without Steplib	OPT.JCL.GEN_JCL.NoStepsWithoutSteplib	NoStepsWithoutSteplib: Every step must have declared a STEPLIB
High	Steplib Instead Of Joblib	OPT.JCL.GEN_JCL.SteplibInsteadOfJoblib	SteplibInsteadOfJoblib: Suggest using local declaration instead of global
High	Allowed Programs	OPT.JCL.MAN_JCL.AllowedPrograms	AllowedPrograms: Allowed programs
High	Deprecated Programs	OPT.JCL.MAN_JCL.DeprecatedPrograms	DeprecatedPrograms: Deprecated programs
High	Job Naming Convention	OPT.JCL.NAM_JCL.JobNamingConvention	JobNamingConvention: JOB statement must comply the specifications
High	Avoid Forbidden Dd Statements	OPT.JCL.PB_JCL.AvoidForbiddenDdStatements	AvoidForbiddenDdStatements: SYSDDTERM, SYSDDDUMP, SYSABEND, SORTLIB are DD statements not allowed
High	Multivolume For Large Files	OPT.JCL.PF_JCL.MultivolumeForLargeFiles	MultivolumeForLargeFiles: Large files must be multivolume
Info	Region Parameter	OPT.JCL.GEN_JCL.RegionParameter	RegionParameter: REGION parameter must have a specific value in JOB, and must not be included in the steps
Info	Dd Sysin Columning	OPT.JCL.MAN_JCL.DdSysinColumning	DdSysinColumning: Every line of the SYSIN DD and SYSTSIN DD text must begin in the specified column
Info	First Step Naming Convention	OPT.JCL.NAM_JCL.FirstStepNamingConvention	FirstStepNamingConvention: The name of the first step must be the specified one
Info	Last Step Naming Convention	OPT.JCL.NAM_JCL.LastStepNamingConvention	LastStepNamingConvention: The name of the last step must be the specified one
Info	Procedures Naming Convention	OPT.JCL.NAM_JCL.ProceduresNamingConvention	ProceduresNamingConvention: Procedure names must comply the pattern
Info	Steps Naming Convention	OPT.JCL.NAM_JCL.StepsNamingConvention	StepsNamingConvention: Step names must have a specific format

Severity	Contrast Rule	Engine rule ID	Description
Low	Avoid Too Long Sysin	OPT.JCL.AvoidTooLongSysin	AvoidTooLongSysin: Avoid a too long SYSIN DD statement
Low	Unit Assignment	OPT.JCL.GEN_JCL.UnitAssignment	UnitAssignment: The UNIT parameter value must be the specified one
Low	One Parameter Per Line In DD	OPT.JCL.MAN_JCL.OneParameterPerLineInDD	OneParameterPerLineInDD: Place every DD parameter in different lines
Low	Files Naming Convention	OPT.JCL.NAM_JCL.FilesNamingConvention	FilesNamingConvention: Checks that file names comply with the patterns
Low	Sort Files Naming Convention	OPT.JCL.NAM_JCL.SortFilesNamingConvention	SortFilesNamingConvention: SORT program files definition format
Low	Step Numbering Convention	OPT.JCL.NAM_JCL.StepNumberingConvention	StepNumberingConvention: Checks if step names comply with specified format
Medium	Avoid Formatting Data After Sorting	OPT.JCL.AvoidFormattingDataAfterSorting	AvoidFormattingDataAfterSorting: Avoid formatting data after sorting, format it before sorting instead
Medium	Number Of Steps	OPT.JCL.MAN_JCL.NumberOfSteps	NumberOfSteps: Limit the number of steps per job
Medium	Only Allowed Includes	OPT.JCL.MAN_JCL.OnlyAllowedIncludes	OnlyAllowedIncludes: Only use the specified include groups
Medium	No Sys1 Prefixed Libs	OPT.JCL.PB_JCL.NoSys1PrefixedLibs	NoSys1PrefixedLibs: Do not use SYS1 prefixed libraries

JSP Scan rules

Contrast Scan supports these rules for JSP.

Severity	Contrast rule	Engine rule ID	Description
Critical	Expression Language Injection	OPT.JSP.SEC_JSP.ExpressionLanguageInjection	ExpressionLanguageInjection: Expression Language (EL / OGNL) injection
Critical	Long JavaScript scripts	OPT.JSP.CFFJSP.ALJS	ALJS: Avoid long js scripts
Critical	Duplicate pages in multiple locations	OPT.JSP.CFFJSP.DPNM	DPNM: There are duplicate pages in different locations
Critical	JSP Page Name	OPT.JSP.CFFJSP.JSPPageName	JSPPageName: The name of the jsp page should follow a naming standard
Critical	JSP Forward found	OPT.JSP.CFFJSP.NJFW	NJFW: There are direct invocations to other jsp pages
Critical	Long script code	OPT.JSP.CFFJSP.NLSC	NLSC: Avoid long scriptlets
Critical	JSP files not in the configured folder	OPT.JSP.CFFJSP.UECD	UECD: JSP pages located in a folder different from the configured folder
Critical	Don't Mix Jstl And Jsf Tags	OPT.JSP.PB_JSF.DontMixJstlAndJsfTags	DontMixJstlAndJsfTags: Nesting JSTL tags within JSF tags, or vice versa
Critical	Don't Use Conditional Tags In Iterative Tags	OPT.JSP.PB_JSF.DontUseConditionalTagsInIterativeTags	DontUseConditionalTagsInIterativeTags: Do not use conditional tags inside iterative tags

Severity	Contrast rule	Engine rule ID	Description
Critical	Avoid Architecture Classes From JSP Rule	OPT.JSP.SEC_JSP.AvoidArchitectureClassesFromJSPRule	AvoidArchitectureClassesFromJSPRule: JSP pages should not import architecture classes
Critical	File Inclusion Vulnerability	OPT.JSP.SEC_JSP.FileInclusionVulnerability	FileInclusionVulnerability: JSP File Inclusion vulnerability
Critical	Database access from a JSP page	OPT.JSP.CFFJSP.ABDP	ABDP: Database access from a JSP page
Critical	Missing Password Field Masking	OPT.JSP.SEC_JSP.MissingPasswordFieldMasking	MissingPasswordFieldMasking: Password input field is not masked
Critical	Unprotected Transport Credential	OPT.JSP.SEC_JSP.UnprotectedTransportCredential	UnprotectedTransportCredential: Unprotected transport of credentials
High	Path Relative Stylesheet Import	OPT.JSP.SEC_JSP.PathRelativeStylesheetImport	PathRelativeStylesheetImport: Path-Relative Stylesheet Import.
High	Target Blank Vulnerability	OPT.JSP.SEC_JSP.TargetBlankVulnerability	TargetBlankVulnerability: Improper Neutralization of links to external sites
High	Unnecessary spaces, tab and line terminators	OPT.JSP.CFFJSP.AWLL	AWLL: The page contains many unnecessary spaces, tabs and line terminators overall inside loops
High	Check URL	OPT.JSP.CFFJSP.CheckURL	CheckURL: Check the URL in the JSP page
High	Missing comments in JSP pages	OPT.JSP.CFFJSP.ICPJ	ICPJ: There are not any comment in the JSP page
High	Multiple CSS tags	OPT.JSP.CFFJSP.NAEE	NAEE: There are many css tags in the elements included in a loop
High	JavaScript code in JSP pages	OPT.JSP.CFFJSP.NFJS	NFJS: Use of Javascript code in pages
High	Java source code found	OPT.JSP.CFFJSP.NSCR	NSCR: There is JAVA source code in some pages
High	Use of document.write	OPT.JSP.CFFJSP.NUSDW	NUSDW: Insertion of HTML code from JSP file including document.write commands
High	Commented out JavaScript code	OPT.JSP.CFFJSP.NUSJSC	NUSJSC: There are commented lines of javascript
High	Allowed Uris	OPT.JSP.GEN_JSF.AllowedUris	AllowedUris: Taglib declarations must include standard URI's
High	Information Exposure In Get Request	OPT.JSP.SEC_JSP.InformationExposureInGetRequest	InformationExposureInGetRequest: Information exposure through strings sent by GET
Info	Duplicate imports	OPT.JSP.CFFJSP.DJIM	DJIM: Do not put duplicate imports in the JSP files
Low	Form Without Captcha	OPT.JSP.SEC_JSP.FormWithoutCaptcha	FormWithoutCaptcha: Form without CAPTCHA
Low	Managed Beans Naming Convention	OPT.JSP.NAM_JSF.ManagedBeansNamingConvention	ManagedBeansNamingConvention: managed-bean names must comply the pattern
Low	IFrames missing src attribute	OPT.JSP.CFFJSP.IMSA	IMSA: Do not use iframes without src attribute
Medium	Specify Integrity Attribute	OPT.JSP.SEC_JSP.SpecifyIntegrityAttribute	SpecifyIntegrityAttribute: Specify a integrity attribute on the <script> and <link> elements
Medium	Literals in JSP pages	OPT.JSP.CFFJSP.ELED	ELED: There are literals in the JSP page

Severity	Contrast rule	Engine rule ID	Description
Medium	Empty pages	OPT.JSP.CFFJSP.ISEM	ISEM: There are empty pages
Medium	Class attribute found	OPT.JSP.CFFJSP.NCAT	NCAT: No class attribute
Medium	HTML commented code	OPT.JSP.CFFJSP.NHMC	NHMC: HTML commented code
Medium	Inline style found	OPT.JSP.CFFJSP.NISI	NISI: There is no style information in JSP
Medium	Use of repeated data in option lists	OPT.JSP.CFFJSP.NUSO	NUSO: Use combo list with static information, so that you will not need option lists
Medium	No header comments	OPT.JSP.CFFJSP.UHCP	UHCP: JSP page does not have header comments

Kotlin Scan rules

Contrast Scan supports these rules for Kotlin.

Severity	Contrast rule	Engine rule ID	Description
Critical	Insecure SSL	OPT.KOTLIN.SEC.InsecureSSL	InsecureSSL: Insecure SSL configuration
Critical	Too Much Origins Allowed	OPT.KOTLIN.SEC.TooMuchOriginsAllowed	TooMuchOriginsAllowedRule: CORS policy (Cross-origin resource sharing) too broad
Critical	Dynamically Loading Code	OPT.KOTLIN.ANDROID.DynamicallyLoadingCode	DynamicallyLoadingCode: Discourage dynamically loading code
Critical	Intent Manipulation	OPT.KOTLIN.ANDROID.IntentManipulation	IntentManipulation: Intent Manipulation
Critical	Javascript Enabled	OPT.KOTLIN.ANDROID.JavascriptEnabled	JavascriptEnabled: Enabling JavaScript is not recommended
Critical	Javascript Interface Annotation	OPT.KOTLIN.ANDROID.JavascriptInterfaceAnnotation	JavascriptInterfaceAnnotation: Potential code injection via WebView.addJavaScriptInterface()
Critical	Code Injection	OPT.KOTLIN.SEC.CodeInjection	CodeInjectionRule: Dynamic code injection in scripting API
Critical	Command Injection	OPT.KOTLIN.SEC.CommandInjection	CommandInjectionRule: Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
Critical	Connection String Parameter Pollution	OPT.KOTLIN.SEC.ConnectionStringParameterPollution	ConnectionStringParameterPollution: Connection string polluted with untrusted input
Critical	Cross Site Scripting	OPT.KOTLIN.SEC.CrossSiteScripting	CrossSiteScriptingRule: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
Critical	Http Splitting	OPT.KOTLIN.SEC.HttpSplitting	HttpSplittingRule: Improper Neutralization of CRLF Sequences in HTTP Headers ('HTTP Response Splitting')
Critical	Ldap Injection	OPT.KOTLIN.SEC.LdapInjection	LdapInjectionRule: Avoid non-neutralized user-controlled input in LDAP search filters
Critical	Mail Command Injection	OPT.KOTLIN.SEC.MailCommandInjection	MailCommandInjection: Mail Command Injection
Critical	No SQL Injection	OPT.KOTLIN.SEC.NoSQLInjection	NoSQLInjection: Improper neutralization of special elements in data query logic (NoSQL injection)
Critical	Process Control	OPT.KOTLIN.SEC.ProcessControl	ProcessControlRule: Library loaded from untrusted source
Critical	Regex Injection	OPT.KOTLIN.SEC.RegexInjection	RegexInjectionRule: Prevent denial of service attack through malicious regular expression ('Regex Injection')

Severity	Contrast rule	Engine rule ID	Description
Critical	Same Origin Method Execution	OPT.KOTLIN.SEC.SameOriginMethodExecution	SameOriginMethodExecution: Same Origin Method Execution (SOME)
Critical	Server Side Request Forgery	OPT.KOTLIN.SEC.ServerSideRequestForgery	ServerSideRequestForgeryRule: Server-Side Request Forgery (SSRF)
Critical	SQL Injection	OPT.KOTLIN.SEC.SqlInjection	SqlInjectionRule: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
Critical	Xml Entity Injection	OPT.KOTLIN.SEC.XmlEntityInjection	XmlEntityInjectionRule: XML entity injection
Critical	Privilege Escalation Attack	OPT.KOTLIN.ANDROID.PrivilegeEscalationAttack	PrivilegeEscalationAttack: Don't allow applications to execute code using other applications privileges
Critical	Garbage Collector Call	OPT.KOTLIN.GarbageCollectorCall	GarbageCollectorCall: Avoid invoking the garbage collector
Critical	Accessibility Subversion	OPT.KOTLIN.SEC.AccessibilitySubversion	AccessibilitySubversionRule: Java access restriction subverted (Reflection)
Critical	Anonymous Ldap Bind	OPT.KOTLIN.SEC.AnonymousLdapBind	AnonymousLdapBindRule: Access Control - Anonymous LDAP Bind
Critical	Native Code Exposed	OPT.KOTLIN.SEC.NativeCodeExposed	NativeCodeExposed: Native Code Exposed.
Critical	Path Traversal	OPT.KOTLIN.SEC.PathTraversal	PathTraversalRule: Avoid non-neutralized user-controlled input composed in a pathname to a resource
Critical	Android Sticky Broadcast	OPT.KOTLIN.ANDROID.AndroidStickyBroadcast	AndroidStickyBroadcast: Avoid Sticky Broadcasts
Critical	SMS Monitoring	OPT.KOTLIN.ANDROID.SMSMonitoring	SMSMonitoring: Don't use SMS for data input or command
Critical	Password In Redirect	OPT.KOTLIN.SEC.PasswordInRedirect	PasswordInRedirect: Password Management - Password in Redirect
Critical	Hardcoded Crypto Key	OPT.KOTLIN.SEC.HardcodedCryptoKey	HardcodedCryptoKey: Hardcoded cryptographic keys
Critical	Non Random IV With CBC Mode	OPT.KOTLIN.SEC.NonRandomIVWithCBCMode	NonRandomIVWithCBCMode: Not using a Random IV with CBC Mode
Critical	Weak Cryptographic Hash	OPT.KOTLIN.SEC.WeakCryptographicHash	WeakCryptographicHashRule: Weak cryptographic hash
Critical	Weak Encryption	OPT.KOTLIN.SEC.WeakEncryption	WeakEncryptionRule: Weak symmetric encryption algorithm
High	Prevent Backup Vulnerability	OPT.KOTLIN.ANDROID.PreventBackupVulnerability	PreventBackupVulnerability: Inadequate backup configuration
High	Insufficient Session Expiration	OPT.KOTLIN.SEC.InsufficientSessionExpiration	InsufficientSessionExpirationRule: Checks that session expiration interval is positive and does not exceed a limit
High	Web Xml Security Misconfigurations	OPT.KOTLIN.SEC.WebXmlSecurityMisconfigurations	WebXmlSecurityMisconfigurationsRule: Avoid misconfiguring security properties in web.xml descriptor
High	Cross Site Request Forgery	OPT.KOTLIN.SEC.CrossSiteRequestForgery	CrossSiteRequestForgeryRule: Cross-site request forgery (CSRF)
High	External Control Of Configuration Setting	OPT.KOTLIN.SEC.ExternalControlOfConfigurationSetting	ExternalControlOfConfigurationSetting: External Control of System or Configuration Setting
High	Http Parameter Pollution	OPT.KOTLIN.SEC.HttpParameterPollution	HttpParameterPollutionRule: HTTP parameter pollution (HPP)
High	JSON Injection	OPT.KOTLIN.SEC.JSONInjection	JSONInjection: Avoid using non-neutralized user-controlled input into JSON entities - JSON Injection
High	Log Forging	OPT.KOTLIN.SEC.LogForging	LogForging: Improper Output Neutralization for Logs
High	Open Redirect	OPT.KOTLIN.SEC.OpenRedirect	OpenRedirectRule: URL Redirection to Untrusted Site ('Open Redirect')

Severity	Contrast rule	Engine rule ID	Description
High	Resource Injection	OPT.KOTLIN.SEC.ResourceInjection	ResourceInjection: Improper control of resource identifiers ("Resource Injection")
High	Trust Boundary Violation	OPT.KOTLIN.SEC.TrustBoundaryViolation	TrustBoundaryViolationRule: Trust boundary violation
High	Unsafe Reflection	OPT.KOTLIN.SEC.UnsafeReflection	UnsafeReflection: Use of Externally-Controlled Input to Select Classes or Code ('Unsafe Reflection')
High	XPath Injection	OPT.KOTLIN.SEC.XPathInjection	XPathInjectionRule: Improper Neutralization of Data within XPath Expressions ('XPath Injection')
High	Xslt Injection	OPT.KOTLIN.SEC.XsltInjection	XsltInjection: XML Injection (aka Blind XPath Injection)
High	Iterator Has Next Calls Next	OPT.KOTLIN.IteratorHasNextCallsNext	IteratorHasNextCallsNext: Iterator hasNext() calls next().
High	Cookies In Security Decision	OPT.KOTLIN.SEC.CookiesInSecurityDecision	CookiesInSecurityDecision: Reliance on Cookies without Validation and Integrity Checking in a Security Decision
High	Security Check In Overridable Method	OPT.KOTLIN.SEC.SecurityCheckInOverridableMethod	SecurityCheckInOverridableMethodRule: Methods that perform a security check must be declared private or final
High	Spring No Anti Xss Configuration	OPT.KOTLIN.SEC.SpringNoAntiXssConfiguration	SpringNoAntiXssConfiguration: Use defaultHtmlEscape {'OWASP-2021': ['A5'], 'WASC': ['08'], 'PCI-DSS': ['6.5.7'], 'ASVS-v4.0.2': ['3.4.5']}
High	Unhandled SSL Exception	OPT.KOTLIN.SEC.UnhandledSSLException	UnhandledSSLExceptionRule: Unhandled SSL exception
High	User Controlled SQL Primary Key	OPT.KOTLIN.SEC.UserControlledSQLPrimaryKey	UserControlledSQLPrimaryKey: Avoid using an user controlled Primary Key into a query
High	Unpaired Equals Hash Code	OPT.KOTLIN.UnpairedEqualsHashCode	UnpairedEqualsHashCode: Object Model Violation: Just one of equals and hashCode defined.
High	Wrong Equals Signature	OPT.KOTLIN.WrongEqualsSignature	WrongEqualsSignature: Wrong equals() signature.
High	Hardcoded Ip	OPT.KOTLIN.SEC.HardcodedIp	HardcodedIp: Do not write IP address in source code
High	Hardcoded Salt	OPT.KOTLIN.SEC.HardcodedSalt	HardcodedSaltRule: A hardcoded salt can compromise system security
High	Inadequate Padding	OPT.KOTLIN.SEC.InadequatePadding	InadequatePaddingRule: Inadequate padding
High	Insecure Randomness	OPT.KOTLIN.SEC.InsecureRandomness	InsecureRandomnessRule: Standard pseudo-random number generators cannot withstand cryptographic attacks
High	Insecure Transport	OPT.KOTLIN.SEC.InsecureTransport	InsecureTransport: Insecure transport
High	Insufficient Key Size	OPT.KOTLIN.SEC.InsufficientKeySize	InsufficientKeySizeRule: Weak cryptography, insufficient key length
Low	Bad Exception Handling	OPT.KOTLIN.BadExceptionHandling	BadExceptionHandling: Bad exception handling.
Low	Complex Condition	OPT.KOTLIN.ComplexCondition	ComplexCondition: Too complex boolean condition.
Low	Complex Function	OPT.KOTLIN.ComplexFunction	ComplexFunction: Too complex function.
Low	Empty Function	OPT.KOTLIN.EmptyFunction	EmptyFunction: Empty function.
Low	Generic Array Of Primitives	OPT.KOTLIN.GenericArrayOfPrimitives	GenericArrayOfPrimitives: Generic Array of primitive type.
Low	Long Function	OPT.KOTLIN.LongFunction	LongFunction: Long function.
Low	Spread Operator	OPT.KOTLIN.SpreadOperator	SpreadOperator: Use of spread (*) operator.
Low	Too Many Parameters	OPT.KOTLIN.TooManyParameters	TooManyParameters: Too many parameters in function.
Low	Unsafe Cast	OPT.KOTLIN.UnsafeCast	UnsafeCast: Unsafe cast.

Severity	Contrast rule	Engine rule ID	Description
Low	Unused Function Parameter	OPT.KOTLIN.UnusedFunctionParameter	UnusedFunctionParameter: Unused function parameter.
Medium	Plaintext Storage In A Cookie	OPT.KOTLIN.SEC.PlaintextStorageInACookie	PlaintextStorageInACookieRule: Cleartext Storage of Sensitive Information in a Cookie
Medium	Unsafe Cookie	OPT.KOTLIN.SEC.UnsafeCookie	UnsafeCookie: Generate server-side cookies with adequate security properties
Medium	Exported Preference Activity	OPT.KOTLIN.ANDROID.ExportedPreferenceActivity	ExportedPreferenceActivity: Activities extending PreferenceActivity should not be exported
Medium	Avoid Host Name Checks	OPT.KOTLIN.SEC.AvoidHostNameChecks	AvoidHostNameChecksRule: Avoid checks on client-side hostname, that are not reliable due to DNS poisoning
Medium	Format String Injection	OPT.KOTLIN.SEC.FormatStringInjection	FormatStringInjectionRule: Exclude unsanitized user input from format strings
Medium	Serialization Injection	OPT.KOTLIN.SEC.SerializationInjection	SerializationInjection: Deserialization of untrusted data
Medium	Check External Storage Permission	OPT.KOTLIN.ANDROID.CheckExternalStoragePermission	CheckExternalStoragePermission: Check permission usage conformance (External Storage Permission)
Medium	Check Internet Permission	OPT.KOTLIN.ANDROID.CheckInternetPermission	CheckInternetPermission: Check permission usage conformance (Internet Permission)
Medium	Check Location Permission	OPT.KOTLIN.ANDROID.CheckLocationPermission	CheckLocationPermission: Check permission usage conformance (Location Permission)
Medium	Complex Interface	OPT.KOTLIN.ComplexInterface	ComplexInterface: Too complex interface.
Medium	Excessive Method Overloading	OPT.KOTLIN.ExcessiveMethodOverloading	ExcessiveMethodOverloading: Excessive method overloading.
Medium	Excessive Nesting Depth	OPT.KOTLIN.ExcessiveNestingDepth	ExcessiveNestingDepth: Excessive nesting depth.
Medium	For Each On Range	OPT.KOTLIN.ForEachOnRange	ForEachOnRange: ForEach on range.
Medium	Missing When Case	OPT.KOTLIN.MissingWhenCase	MissingWhenCase: Missing when case.
Medium	Detail Error Leak	OPT.KOTLIN.SEC.DetailErrorLeak	DetailErrorLeakRule: Do not send detail error information to client
Medium	Execution After Redirect	OPT.KOTLIN.SEC.ExecutionAfterRedirect	ExecutionAfterRedirect: Execution After Redirect (EAR)
Medium	Too Many Functions	OPT.KOTLIN.TooManyFunctions	TooManyFunctions: Too many functions.
Medium	Unconditional Jump In Loop	OPT.KOTLIN.UnconditionalJumpInLoop	UnconditionalJumpInLoop: Unconditional jump in loop.
Medium	Unreachable Code	OPT.KOTLIN.UnreachableCode	UnreachableCode: Unreachable ("dead") code.
Medium	Unused Private Function	OPT.KOTLIN.UnusedPrivateFunction	UnusedPrivateFunction: Unused private function.
Medium	Hardcoded Username Password	OPT.KOTLIN.SEC.HardcodedUsernamePassword	HardcodedUsernamePassword: Use of Hard-coded Credentials
Medium	JSON P Hijacking	OPT.KOTLIN.SEC.JSONPHijacking	JSONPHijacking: Sensitive information exposed through JSONP
Medium	Password In Configuration File	OPT.KOTLIN.SEC.PasswordInConfigurationFile	PasswordInConfigurationFile: Use of credentials into configuration file
Medium	Plaintext Storage Of Password	OPT.KOTLIN.SEC.PlaintextStorageOfPassword	PlaintextStorageOfPassword: Plaintext Storage of a Password
Medium	Privacy Violation	OPT.KOTLIN.SEC.PrivacyViolation	PrivacyViolation: Exposure of Private Information ('Privacy Violation')

Severity	Contrast rule	Engine rule ID	Description
Medium	Serializable Class Containing Sensitive Data	OPT.KOTLIN.SEC.SerializableClassContainingSensitiveData	SerializableClassContainingSensitiveData: Serializable Class Containing Sensitive Data

NATURAL Scan rules

Contrast Scan supports these rules for NATURAL.

Severity	Contrast rule	Engine rule ID	Description
Critical	Avoid Input Using Map Inside Read Find	OPT.NATURAL.NAT_PF.AvoidInputUsingMapInsideReadFind	AvoidInputUsingMapInsideReadFind: Avoid records locking for user input
High	Avoid Too Complex Routines	OPT.NATURAL.NAT_MAN.AvoidTooComplexRoutines	AvoidTooComplexRoutines: Avoid too complex programmes (in cyclomatic complexity, number of different execution ways)
High	Only Allowed Copy Codes	OPT.NATURAL.NAT_MAN.OnlyAllowedCopyCodes	OnlyAllowedCopyCodes: Limit the copy codes to be used
High	Unused Data Area	OPT.NATURAL.NAT_MAN.UnusedDataArea	UnusedDataArea: Avoid unused data areas (local, global or parameter) in system
High	Unused Maps	OPT.NATURAL.NAT_MAN.UnusedMaps	UnusedMaps: Avoid unused maps / forms declarations in system
High	Unused Routines	OPT.NATURAL.NAT_MAN.UnusedRoutines	UnusedRoutines: Avoid unused routines or programs in system
High	Avoid D M L Out Of Transaction Context	OPT.NATURAL.NAT_PB.AvoidDMLOutOfTransactionContext	AvoidDMLOutOfTransactionContext: Check that every DML statement has its corresponding END TRANSACTION
High	Avoid Empty None Clause	OPT.NATURAL.NAT_PB.AvoidEmptyNoneClause	AvoidEmptyNoneClause: Avoid empty NONE clauses in the DECIDE ON/FOR statement in programmes
High	Avoid Empty On Error Clause	OPT.NATURAL.NAT_PB.AvoidEmptyOnErrorClause	AvoidEmptyOnErrorClause: It's advisable to not code empty ON ERROR to handle correctly every possible error
High	Avoid Stack Usages	OPT.NATURAL.NAT_PB.AvoidStackUsages	AvoidStackUsages: Avoid using STACK sentence
High	Avoid Find Sorted By	OPT.NATURAL.NAT_PF.AvoidFindSortedBy	AvoidFindSortedBy: Do not use option SORTED BY in FIND statements
High	Use With Limit Clause In Read And Find	OPT.NATURAL.NAT_PF.UseWithLimitClauseInReadAndFind	UseWithLimitClauseInReadAndFind: Limit the amount of records used in READ and FIND
Info	Avoid Multiple Sentences In One Line	OPT.NATURAL.NAT_DOC.AvoidMultipleSentencesInOneLine	AvoidMultipleSentencesInOneLine: Avoid writing multiple sentences in one code line
Info	Avoid Too Large Code Blocks	OPT.NATURAL.NAT_MAN.AvoidTooLargeCodeBlocks	AvoidTooLargeCodeBlocks: Certain code structures (LDA, PDA, programs or subroutines) should not exceed a maximum number of lines
Info	Remove Commented Code	OPT.NATURAL.NAT_MAN.RemoveCommentedCode	RemoveCommentedCode: Avoid commented code
Info	Program Name	OPT.NATURAL.NAT_NOM.ProgramName	ProgramName: Follow programmes naming standard

Severity	Contrast rule	Engine rule ID	Description
Info	Subroutine Name	OPT.NATURAL.NAT_NOM.SubroutineName	SubroutineName: Follow subroutines naming standard
Info	Variable Name	OPT.NATURAL.NAT_NOM.VariableName	VariableName: Follow variables naming standard
Info	Avoid Escape Top	OPT.NATURAL.NAT_PB.AvoidEscapeTop	AvoidEscapeTop: Do not use ESCAPE TOP statement
Info	Avoid Debugging Write In Online Progs	OPT.NATURAL.NAT_MAN.AvoidDebuggingWriteInOnlineProgs	AvoidDebuggingWriteInOnlineProgs: Avoid debugging WRITE in online-processing programs.
Low	Avoid Excessive Record Nesting	OPT.NATURAL.NAT_MAN.AvoidExcessiveRecordNesting	AvoidExcessiveRecordNesting: Avoid variable declarations (records) too deeply nested
Low	Number Of Options In Decide	OPT.NATURAL.NAT_MAN.NumberOfOptionsInDecide	NumberOfOptionsInDecide: Try to limit the number of options in DECIDE statements
Low	Avoid Find Read With Hold	OPT.NATURAL.NAT_PF.AvoidFindReadWithHold	AvoidFindReadWithHold: Detects READ/FIND sentences with UPDATE/STORE/DELETE instructions in its body
Low	Avoid Where In Read And Find Statement	OPT.NATURAL.NAT_PF.AvoidWhereInReadAndFindStatement	AvoidWhereInReadAndFindStatement: Avoid WHERE clause in READ and FIND
Medium	Avoid Deep Nesting Branches	OPT.NATURAL.NAT_MAN.AvoidDeepNestingBranches	AvoidDeepNestingBranches: Avoid deep nesting in flow control sentences (IF, FOR, REPEAT, DECIDE FOR, DECIDE ON)
Medium	Avoid Accept Reject If	OPT.NATURAL.NAT_PB.AvoidAcceptRejectIf	AvoidAcceptRejectIf: Do not use ACCEPT/REJECT IF statement
Medium	Avoid Recursive Calls	OPT.NATURAL.NAT_PB.AvoidRecursiveCalls	AvoidRecursiveCalls: Avoid recursive calls (PERFORM to the same subroutine , CALLNAT ... RUN to the same program/subprogram)
Medium	Avoid Stop Statement	OPT.NATURAL.NAT_PB.AvoidStopStatement	AvoidStopStatement: Avoid STOP statement and use TERMINATE with the corresponding error code instead
Medium	Avoid Terminate Without Error Code	OPT.NATURAL.NAT_PB.AvoidTerminateWithoutErrorCode	AvoidTerminateWithoutErrorCode: It's advisable to code TERMINATE with its corresponding error code
Medium	End Transactions In Program Body	OPT.NATURAL.NAT_PB.EndTransactionsInProgramBody	EndTransactionsInProgramBody: Avoid placing BACKOUT or END TRANSACTION out of the programme body
Medium	Only One Exit Point	OPT.NATURAL.NAT_PB.OnlyOneExitPoint	OnlyOneExitPoint: Code only one exit point to avoid possible errors
Medium	Avoid Move By Name	OPT.NATURAL.NAT_PF.AvoidMoveByName	AvoidMoveByName: It is better to move based on the structure of the group or redefined variable
Medium	Avoid Sort	OPT.NATURAL.NAT_PF.AvoidSort	AvoidSort: Do not use SORT statement

Objective-C Scan rules

Contrast Scan supports these rules for Objective-C.

Severity	Contrast rule	Engine rule ID	Description
Critical	Avoid SQL Injection	OPT.OBJECTIVEC.AvoidSqlInjection	AvoidSqlInjection: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
Critical	Code Injection	OPT.OBJECTIVEC.CodeInjection	CodeInjection: Improper Control of Generation Code ('Code Injection')
Critical	Cross Site Scripting	OPT.OBJECTIVEC.CrossSiteScripting	CrossSiteScripting: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
Critical	DoS Regular Expression	OPT.OBJECTIVEC.DoSRegularExpression	DoSRegularExpression: Prevent denial of service attack through malicious regular expression
Critical	Format String Vulnerability	OPT.OBJECTIVEC.FormatStringVulnerability	FormatStringVulnerability: Exclude unsanitized user input from format strings
Critical	JSON Injection	OPT.OBJECTIVEC.JSONInjection	JSONInjection: Avoid using non-neutralized uncontrolled input into JSON entities - JSON Injection
Critical	Open Redirect	OPT.OBJECTIVEC.OpenRedirect	OpenRedirect: URL Redirection to Untrusted Site ('Open Redirect')
Critical	XML Entity Injection	OPT.OBJECTIVEC.XMLEntityInjection	XMLEntityInjection: XML entity injection
Critical	XPath Injection	OPT.OBJECTIVEC.XPathInjection	XPathInjection: Improper Neutralization of Data within XPath Expressions ('XPath Injection')
Critical	Command Injection Rule	OPT.OBJECTIVEC.SECURITY.CommandInjectionRule	CommandInjectionRule: Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
Critical	Connection String Parameter Pollution	OPT.OBJECTIVEC.SECURITY.ConnectionStringParameterPollution	ConnectionStringParameterPollution: Connection string polluted with untrusted input
Critical	Http Splitting Rule	OPT.OBJECTIVEC.SECURITY.HttpSplittingRule	HttpSplittingRule: Improper Neutralization of C Sequences in HTTP Headers ('HTTP Response Splitting')
Critical	Mail Command Injection	OPT.OBJECTIVEC.SECURITY.MailCommandInjection	MailCommandInjection: Mail Command Injection
Critical	No SQL Injection	OPT.OBJECTIVEC.SECURITY.NoSQLInjection	NoSQLInjection: Improper neutralization of special elements in data query logic (NoSQL injection)
Critical	Avoid Confusing User Id Calls	OPT.OBJECTIVEC.AvoidConfusingUserIdCalls	AvoidConfusingUserIdCalls: Avoid setuid() / setreuid() / setgid() / setregid() to change process privilege levels
Critical	Avoid Empty Catch Blocks	OPT.OBJECTIVEC.AvoidEmptyCatchBlocks	AvoidEmptyCatchBlocks: Avoid empty @catch blocks
Critical	Avoid Large Methods	OPT.OBJECTIVEC.AvoidLargeMethods	AvoidLargeMethods: Avoid methods with excessive number of lines
Critical	Avoid Loop With Empty Body	OPT.OBJECTIVEC.AvoidLoopWithEmptyBody	AvoidLoopWithEmptyBody: Avoid loops (while, while, for) with empty body
Critical	Avoid Sudo	OPT.OBJECTIVEC.AvoidSudo	AvoidSudo: Avoid using sudo programmatically
Critical	Avoid Throwing Exceptions	OPT.OBJECTIVEC.AvoidThrowingExceptions	AvoidThrowingExceptions: Avoid throwing exceptions
Critical	Nil In Literals	OPT.OBJECTIVEC.NilInLiterals	NilInLiterals: Do not use nil in NSArray or NSDictionary literals
Critical	No Update Loop Vars In For Body	OPT.OBJECTIVEC.NoUpdateLoopVarsInForBody	NoUpdateLoopVarsInForBody: Do not update control vars in 'for' loop body
Critical	Override Draw Rect UIView Subclasses	OPT.OBJECTIVEC.OverrideDrawRectUIViewSubclasses	OverrideDrawRectUIViewSubclasses: Call super when overriding drawRect: and superclass is a UIView subclass
Critical	Override Is Equal And Hash	OPT.OBJECTIVEC.OverrideIsEqualAndHash	OverrideIsEqualAndHash: Override hash method when overriding isEqual: method

Severity	Contrast rule	Engine rule ID	Description
Critical	Override U I View Controller Methods	OPT.OBJECTIVEC.OverrideUIViewControllerMethods	OverrideUIViewControllerMethods: Call super when overriding some of UIViewController methods
Critical	Override U I View Methods	OPT.OBJECTIVEC.OverrideUIViewMethods	OverrideUIViewMethods: Call super when overriding some of UIView methods
Critical	Path Manipulation Vulnerability	OPT.OBJECTIVEC.PathManipulationVulnerability	PathManipulationVulnerability: Avoid non-neutralized user-controlled input to be part of a pathname (file or directory) used in I/O operations
Critical	Replace With Less Secure Func	OPT.OBJECTIVEC.ReplaceWithLessSecureFunc	ReplaceWithLessSecureFunc: Do not replace secure functions with less secure functions
Critical	Reuse Annotation Views	OPT.OBJECTIVEC.ReuseAnnotationViews	ReuseAnnotationViews: Reuse annotation view maps
Critical	Reuse Table View Cells	OPT.OBJECTIVEC.ReuseTableViewCell	ReuseTableViewCell: Reuse cells in table view
Critical	Missing Password Field Masking	OPT.OBJECTIVEC.SECURITY.MissingPasswordFieldMasking	MissingPasswordFieldMasking: Password input field is not masked
Critical	Certificate Verify Failed Bypass	OPT.OBJECTIVEC.CertificateVerifyFailedBypass	CertificateVerifyFailedBypass: Do not bypass certificate validation fails
Critical	Hardcoded Crypto Key	OPT.OBJECTIVEC.SECURITY.HardcodedCryptoKey	HardcodedCryptoKey: Hardcoded cryptographic keys
Critical	Weak Key Derivation Iteration	OPT.OBJECTIVEC.SECURITY.WeakKeyDerivationIteration	WeakKeyDerivationIteration: Too weak iteration count on key derivation
Critical	Weak Key Derivation Password	OPT.OBJECTIVEC.SECURITY.WeakKeyDerivationPassword	WeakKeyDerivationPassword: Empty or nil password used in key derivation
High	Do Not Use System	OPT.OBJECTIVEC.DoNotUseSystem	DoNotUseSystem: Do not call system() if you do not need a command processor
High	Perform Selector With Untrusted Data	OPT.OBJECTIVEC.PerformSelectorWithUntrustedData	PerformSelectorWithUntrustedData: Avoid external control over performSelector
High	URL Schemes Handling	OPT.OBJECTIVEC.URLSchemesHandling	URLSchemesHandling: Verify invoker application identity
High	Http Parameter Pollution Rule	OPT.OBJECTIVEC.SECURITY.HttpParameterPollutionRule	HttpParameterPollutionRule: HTTP parameter pollution (HPP)
High	Log Forging	OPT.OBJECTIVEC.SECURITY.LogForging	LogForging: Improper Output Neutralization for Logs
High	Resource Injection	OPT.OBJECTIVEC.SECURITY.ResourceInjection	ResourceInjection: Improper control of resource identifiers ("Resource Injection")
High	URL Scheme Hijacking	OPT.OBJECTIVEC.SECURITY.URLSchemeHijacking	URLSchemeHijacking: URL scheme hijacking through user input
High	XML Injection	OPT.OBJECTIVEC.SECURITY.XMLInjection	XMLInjection: XML Injection (aka Blind XPath Injection)
High	Assign Init Result To Self	OPT.OBJECTIVEC.AssignInitResultToSelf	AssignInitResultToSelf: Assign the result of [super init] to self in init methods and check for nil
High	Avoid Conditional Operator	OPT.OBJECTIVEC.AvoidConditionalOperator	AvoidConditionalOperator: Do not use ? ternary operator to evaluate conditions
High	Avoid Empty Draw Rect	OPT.OBJECTIVEC.AvoidEmptyDrawRect	AvoidEmptyDrawRect: Avoid empty drawRect implementations

Severity	Contrast rule	Engine rule ID	Description
High	Avoid Insecure C String Functions	OPT.OBJECTIVEC.AvoidInsecureCStringFunctions	AvoidInsecureCStringFunctions: Avoid C library functions that do not check for bounds
High	Avoid Maximum Location Accuracy When Possible	OPT.OBJECTIVEC.AvoidMaximumLocationAccuracyWhenPossible	AvoidMaximumLocationAccuracyWhenPossible: Avoid using by default the best location accuracy
High	Balance Custom Getters And Setters	OPT.OBJECTIVEC.BalanceCustomGettersAndSetters	BalanceCustomGettersAndSetters: Always write a custom getter for a property where you have a custom setter, and viceversa
High	Boolean In Comparisons	OPT.OBJECTIVEC.BooleanInComparisons	BooleanInComparisons: Avoid using nil/NO or YES in comparisons
High	Cache N S Date Formatters	OPT.OBJECTIVEC.CacheNSDateFormatters	CacheNSDateFormatters: Cache a single instance from NSDateFormatter types instead of creating multiple instances
High	Claim Ownership Core Foundation Objects	OPT.OBJECTIVEC.ClaimOwnershipCoreFoundationObjects	ClaimOwnershipCoreFoundationObjects: Claim ownership of Core Foundation objects received from Core Foundation Get functions
High	Class Cyclomatic Complexity	OPT.OBJECTIVEC.ClassCyclomaticComplexity	ClassCyclomaticComplexity: Avoid using classes with high cyclomatic complexity values
High	Clear Frame Buffers Before Drawing	OPT.OBJECTIVEC.ClearFrameBuffersBeforeDrawing	ClearFrameBuffersBeforeDrawing: Call glClear function before drawing
High	Comment Top Level Declarations	OPT.OBJECTIVEC.CommentTopLevelDeclarations	CommentTopLevelDeclarations: Interfaces, categories and protocols should have an accompanying comment
High	Create Autorelease Pool In Thread	OPT.OBJECTIVEC.CreateAutoreleasePoolInThread	CreateAutoreleasePoolInThread: Create an autorelease pool in each thread
High	Deallocation Of Objects Removed From Collections	OPT.OBJECTIVEC.DeallocationOfObjectsRemovedFromCollections	DeallocationOfObjectsRemovedFromCollections: Avoid deallocation of objects removed from fundamental collection classes (NSMutableArray, NSMutableDictionary) that you are going to use
High	Dealloc Method	OPT.OBJECTIVEC.DeallocMethod	DeallocMethod: Not invoke to the superclass's implementation at the end of the dealloc implementation
High	Default Clause Switch Statements	OPT.OBJECTIVEC.DefaultClauseSwitchStatements	DefaultClauseSwitchStatements: All switch statements must have a default statement
High	Designated_INITIALIZER	OPT.OBJECTIVEC.DesignatedInitializer	DesignatedInitializer: Every public class must have at least one designated initializer
High	Distance From Main Sequence	OPT.OBJECTIVEC.DistanceFromMainSequence	DistanceFromMainSequence: Project should not be too far from main sequence
High	Do Not Instantiate Temporal Objects Loops	OPT.OBJECTIVEC.DoNotInstantiateTemporalObjectsLoops	DoNotInstantiateTemporalObjectsLoops: Avoid instantiating temporal objects in loop bodies
High	Fork Followed By Exec	OPT.OBJECTIVEC.ForkFollowedByExec	ForkFollowedByExec: A call to fork must be followed by a call to exec or a similar function

Severity	Contrast rule	Engine rule ID	Description
High	Handle Memory Warnings	OPT.OBJECTIVEC.HandleMemoryWarnings	HandleMemoryWarnings: Respond to low-memory warnings
High	Many Cases	OPT.OBJECTIVEC.ManyCases	ManyCases: Avoid too many choices in switch structures
High	Method Cyclomatic Complexity	OPT.OBJECTIVEC.MethodCyclomaticComplexity	MethodCyclomaticComplexity: Avoid using methods with high cyclomatic complexity values
High	Minimize Bluetooth Interaction	OPT.OBJECTIVEC.MinimizeBluetoothInteraction	MinimizeBluetoothInteraction: Avoid using CBCentralManagerScanOptionAllowDuplicate as a scan option
High	Nested If Statements	OPT.OBJECTIVEC.NestedIfStatements	NestedIfStatements: Avoid a high level of if statement nesting
High	Notify Deallocation Weak References	OPT.OBJECTIVEC.NotifyDeallocationWeakReferences	NotifyDeallocationWeakReferences: Notify deallocation in weak-referenced objects
High	Low Cohesion Within Object	OPT.OBJECTIVEC.LowCohesionWithinObject	LowCohesionWithinObject: Avoid classes with low degree of cohesion
High	Parenthesize Macro Args	OPT.OBJECTIVEC.ParenthesizeMacroArgs	ParenthesizeMacroArgs: Macro replacement should be parenthesized
High	Property Data Member	OPT.OBJECTIVEC.PropertyDataMember	PropertyDataMember: Create a property for each data member and never access instance variables directly
High	Release Core Foundation Objects	OPT.OBJECTIVEC.ReleaseCoreFoundationObjects	ReleaseCoreFoundationObjects: Relinquish ownership of owned Core Foundation objects
High	Release Ivars Dealloc	OPT.OBJECTIVEC.ReleaselvarsDealloc	ReleaselvarsDealloc: Release ivars for retained copied properties in dealloc method
High	Release Owned Objects	OPT.OBJECTIVEC.ReleaseOwnedObjects	ReleaseOwnedObjects: Release owned objects in MRR
High	Sizeof Pointer Instead Array	OPT.OBJECTIVEC.SizeofPointerInsteadArray	SizeofPointerInsteadArray: Do not apply the sizeof operator to a pointer when taking the size of an array
High	Specify Path For Shadows	OPT.OBJECTIVEC.SpecifyPathForShadows	SpecifyPathForShadows: Specify shadowPath property of layer when drawing a shadow
High	Subviews In Standard Controls	OPT.OBJECTIVEC.SubviewsInStandardControls	SubviewsInStandardControls: Do not add subviews to standard system controls
High	Unstructured Branching Statements	OPT.OBJECTIVEC.UnstructuredBranchingStatements	UnstructuredBranchingStatements: Avoid using unstructured branching statements
High	Unused Local Var	OPT.OBJECTIVEC.UnusedLocalVar	UnusedLocalVar: Avoid unused local variables
High	Unused Method Parameter	OPT.OBJECTIVEC.UnusedMethodParameter	UnusedMethodParameter: Avoid unused method parameters
High	Use Automatic Reference Counting	OPT.OBJECTIVEC.UseAutomaticReferenceCounting	UseAutomaticReferenceCounting: Code must obey transition to ARC rules
High	Use Block Based Animation	OPT.OBJECTIVEC.UseBlockBasedAnimation	UseBlockBasedAnimation: Use block-based animations
High	Use Setter For Property	OPT.OBJECTIVEC.UseSetterForProperty	UseSetterForProperty: When setting property values, use setter method
High	User Controlled SQL Primary Key	OPT.OBJECTIVEC.SECURITY.UserControlledSQLPrimaryKey	UserControlledSQLPrimaryKey: Avoid using a user controlled Primary Key into a query

Severity	Contrast rule	Engine rule ID	Description
High	Insecure Transport Layer	OPT.OBJECTIVEC.InsecureTransportLayer	InsecureTransportLayer: Avoid using HTTP instead of HTTPS
High	Hardcoded Ip	OPT.OBJECTIVEC.SECURITY.HardcodedIp	HardcodedIp: Do not write IP address in source code
High	Weak Cryptographic Hash	OPT.OBJECTIVEC.WeakCryptographicHash	WeakCryptographicHash: Weak cryptographic hashes cannot guarantee data integrity
High	Weak Encryption	OPT.OBJECTIVEC.WeakEncryption	WeakEncryption: Weak encryption algorithm
Low	Avoid CGContext Flush	OPT.OBJECTIVEC.AvoidCGContextFlush	AvoidCGContextFlush: Avoid calling CGContextFlush
Low	Avoid Exposing Instance Vars	OPT.OBJECTIVEC.AvoidExposingInstanceVars	AvoidExposingInstanceVars: Instance variables should be properly hidden
Low	Avoid Function Like Macros	OPT.OBJECTIVEC.AvoidFunctionLikeMacros	AvoidFunctionLikeMacros: AvoidFunctionLikeMacros: Prefer inline or static functions to function-like macros
Low	Avoid Locks	OPT.OBJECTIVEC.AvoidLocks	AvoidLocks: Avoid using locks
Low	Avoid Single Word Titles In Alerts	OPT.OBJECTIVEC.AvoidSingleWordTitlesInAlerts	AvoidSingleWordTitlesInAlerts: Avoid 'Alert' elements with little explanatory title
Low	Category In Framework Class Name Conventions	OPT.OBJECTIVEC.CategoryInFrameworkClassNameConventions	CategoryInFrameworkClassNameConventions: Naming convention for category methods in framework classes
Low	Dead Code	OPT.OBJECTIVEC.DeadCode	DeadCode: Avoid unreachable code
Low	Declare Subviews As Opaque	OPT.OBJECTIVEC.DeclareSubviewsAsOpaque	DeclareSubviewsAsOpaque: Declare subview opaque
Low	Reference From Parent To Child Class	OPT.OBJECTIVEC.ReferenceFromParentToChildClass	ReferenceFromParentToChildClass: Parent class does not reference any of its child classes
Low	Password In Comment Rule	OPT.OBJECTIVEC.SECURITY.PasswordInCommentRule	PasswordInCommentRule: Storing passwords in plaintext anywhere in the system or system code can compromise system security
Medium	Plaintext Storage In A Cookie Rule	OPT.OBJECTIVEC.SECURITY.PlaintextStorageInACookieRule	PlaintextStorageInACookieRule: Cleartext Storage of Sensitive Information in a Cookie
Medium	Unsafe Cookie	OPT.OBJECTIVEC.SECURITY.UnsafeCookie	UnsafeCookie: Generate server-side cookies with adequate security properties
Medium	Serialization Injection	OPT.OBJECTIVEC.SECURITY.SerializationInjection	SerializationInjection: Deserialization of untrusted data
Medium	Avoid Comparing Float Numbers	OPT.OBJECTIVEC.AvoidComparingFloatNumbers	AvoidComparingFloatNumbers: Avoid comparing floating point with Code Quality [] {}
Medium	Avoid Magic Numbers	OPT.OBJECTIVEC.AvoidMagicNumbers	AvoidMagicNumbers: Avoid using numeric constants
Medium	Avoid NSLog	OPT.OBJECTIVEC.AvoidNSLog	AvoidNSLog: Avoid the use of NSLog except in debug mode DEUGB
Medium	Avoid Querying State Open GL ES	OPT.OBJECTIVEC.AvoidQueryingStateOpenGL	AvoidQueryingStateOpenGL: Avoid calls to glGet*() to preserve parallelism
Medium	Avoid Super In Load View	OPT.OBJECTIVEC.AvoidSuperInLoadView	AvoidSuperInLoadView: Avoid calling super without overriding loadView

Severity	Contrast rule	Engine rule ID	Description
Medium	Avoid Too Deep Class Hierarchies	OPT.OBJECTIVEC.AvoidTooDeepClassHierarchies	AvoidTooDeepClassHierarchies: Avoid too deep hierarchy classes
Medium	Avoid Unsafe File Functions	OPT.OBJECTIVEC.AvoidUnsafeFileFunctions	AvoidUnsafeFileFunctions: Use safe file access POSIX functions
Medium	Background Apps Open G L E S Commands	OPT.OBJECTIVEC.BackgroundAppsOpenGLESCommands	BackgroundAppsOpenGLESCommands: Avoid submitting OpenGL ES commands when your app is in background
Medium	Be Aware Of Location Errors	OPT.OBJECTIVEC.BeAwareOfLocationErrors	BeAwareOfLocationErrors: Be aware of location errors
Medium	Break Non Empty Switch Clauses	OPT.OBJECTIVEC.BreakNonEmptySwitchClauses	BreakNonEmptySwitchClauses: Non empty case sentences must end with a break sentence
Medium	Check Parameter Number In Method	OPT.OBJECTIVEC.CheckParameterNumberInMethod	CheckParameterNumberInMethod: Too many parameters in method
Medium	Class Factory Methods Name Convention	OPT.OBJECTIVEC.ClassFactoryMethodsNameConvention	ClassFactoryMethodsNameConvention: Naming convention for class factory methods
Medium	Copy Immutable Objects	OPT.OBJECTIVEC.CopyImmutableObjects	CopyImmutableObjects: Always use the (copy) storage class for properties that receive objects that have mutable subclasses
Medium	Density Of Comments	OPT.OBJECTIVEC.DensityOfComments	DensityOfComments: Source code must be properly commented
Medium	Font Size	OPT.OBJECTIVEC.FontSize	FontSize: Avoid using fonts smaller than 11 points
Medium	High Coupling Between Objects	OPT.OBJECTIVEC.HighCouplingBetweenObjects	HighCouplingBetweenObjects: Classes internally strongly coupled must be avoided
Medium	Maximum Number Of Methods	OPT.OBJECTIVEC.MaximumNumberOfMethods	MaximumNumberOfMethods: Number of methods in an interface or protocol not should exceed a threshold
Medium	Misuse Embedding In Scroll View	OPT.OBJECTIVEC.MisuseEmbeddingInScrollView	MisuseEmbeddingInScrollView: Avoid embedding a UIWebView or UITableView in a UIScrollView
Medium	One Statement Per Line	OPT.OBJECTIVEC.OneStatementPerLine	OneStatementPerLine: Use only one statement per line
Medium	Replace Enum By Ns Enum Or Ns Option	OPT.OBJECTIVEC.ReplaceEnumByNsEnumOrNsOption	ReplaceEnumByNsEnumOrNsOption: Replace enum declarations by uses of NS_ENUM and NS_OPTIONS macros
Medium	Too Many Buttons In Action Sheet	OPT.OBJECTIVEC.TooManyButtonsInActionSheet	TooManyButtonsInActionSheet: Avoid defining many buttons in an Action Sheet
Medium	Too Many Dots In Page Control	OPT.OBJECTIVEC.TooManyDotsInPageControl	TooManyDotsInPageControl: Avoid too many opened views in Page Control
Medium	Touch Controls Size	OPT.OBJECTIVEC.TouchControlsSize	TouchControlsSize: Touch controls must should have at least a 44 x 44 pixels dimension
Medium	Use Instancetype Instead Of Id	OPT.OBJECTIVEC.UseInstancetypeInsteadOfId	UseInstancetypeInsteadOfId: Alloc, init and class factory methods must return instancetype instead of id
Medium	Use Modern File A P I	OPT.OBJECTIVEC.UseModernFileAPI	UseModernFileAPI: Use modern file APIs
Medium	Use Nonatomic Attribute	OPT.OBJECTIVEC.UseNonatomicAttribute	UseNonatomicAttribute: Always use the "nonatomic" attribute on your properties

Severity	Contrast rule	Engine rule ID	Description
Medium	Wrap Macro Statements In Do While	OPT.OBJECTIVEC.WrapMacroStatementsInDoWhile	WrapMacroStatementsInDoWhile: Wrap multistatement macros in a do-while loop
Medium	Avoid S M S	OPT.OBJECTIVEC.SECURITY.AvoidSMS	AvoidSMS: Avoid performing SMS-related operations
Medium	Biometric Without Message	OPT.OBJECTIVEC.SECURITY.BiometricWithoutMessage	BiometricWithoutMessage: User is asked for fingerprints without reason
Medium	Execution After Redirect	OPT.OBJECTIVEC.SECURITY.ExecutionAfterRedirect	ExecutionAfterRedirect: Execution After Redirect (EAR)
Medium	Missing Content Validation	OPT.OBJECTIVEC.SECURITY.MissingContentValidation	MissingContentValidation: Missing Content Validation
Medium	Potential Infinite Loop	OPT.OBJECTIVEC.SECURITY.PotentialInfiniteLoop	PotentialInfiniteLoop: Loop with Unreachable Condition ('Infinite Loop')
Medium	Server Trust Credential Check	OPT.OBJECTIVEC.SECURITY.ServerTrustCredentialCheck	ServerTrustCredentialCheck: Evaluate server certificate trust chain
Medium	Unchecked Input In Loop Condition	OPT.OBJECTIVEC.SECURITY.UncheckedInputInLoopCondition	UncheckedInputInLoopCondition: Unchecked in loop condition
Medium	Cookie Without SSL	OPT.OBJECTIVEC.CookieWithoutSSL	CookieWithoutSSL: Avoid creating cookies with security attributes
Medium	Hardcoded Username Password	OPT.OBJECTIVEC.SECURITY.HardcodedUsernamePassword	HardcodedUsernamePassword: Use of Hard-coded Credentials
Medium	Http Response Caching Leak	OPT.OBJECTIVEC.SECURITY.HttpResponseCachingLeak	HttpResponseCachingLeak: HTTP sensitive responses being cached
Medium	Information Exposure Through Error Message	OPT.OBJECTIVEC.SECURITY.InformationExposureThroughErrorMessage	InformationExposureThroughErrorMessage: A sensitive information exposure through error messages
Medium	Insecure Temporary File	OPT.OBJECTIVEC.SECURITY.InsecureTemporaryFile	InsecureTemporaryFile: Creating and using insecure temporary files can leave application system data vulnerable to attack.
Medium	Keyboard Caching Leak	OPT.OBJECTIVEC.SECURITY.KeyboardCachingLeak	KeyboardCachingLeak: Sensitive data leaked through keyboard cache
Medium	Password In Configuration File	OPT.OBJECTIVEC.SECURITY.PasswordInConfigurationFile	PasswordInConfigurationFile: Use of credentials into configuration file
Medium	Pasteboard Caching Leak	OPT.OBJECTIVEC.SECURITY.PasteboardCachingLeak	PasteboardCachingLeak: Sensitive data leaked through the pasteboard caching mechanism
Medium	Privacy Violation	OPT.OBJECTIVEC.SECURITY.PrivacyViolation	PrivacyViolation: Exposure of Private Information ('Privacy Violation')
Medium	Screen Caching Leak	OPT.OBJECTIVEC.SECURITY.ScreenCachingLeak	ScreenCachingLeak: Sensitive data leaked through the screen caching mechanism when is backgrounded
Medium	Sensitive Core Data	OPT.OBJECTIVEC.SECURITY.SensitiveCoreData	SensitiveCoreData: Sensitive data stored into CoreData('Privacy Violation')
Medium	Sensitive Data Accessed From Itunes	OPT.OBJECTIVEC.SECURITY.SensitiveDataAccessedFromItunes	SensitiveDataAccessedFromItunes: Sensitive data accessed from Itunes ('Privacy Violation')
Medium	Sensitive No SQL	OPT.OBJECTIVEC.SECURITY.SensitiveNoSQL	SensitiveNoSQL: Sensitive data stored into a NoSQL database('Privacy Violation')
Medium	Sensitive SQL	OPT.OBJECTIVEC.SECURITY.SensitiveSQL	SensitiveSQL: Sensitive data stored into a SQL database('Privacy Violation')

Severity	Contrast rule	Engine rule ID	Description
Medium	Sensitive User Defaults	OPT.OBJECTIVEC.SECURITY.SensitiveUserDefaults	SensitiveUserDefaults: Sensitive data stored in NSUserDefaults('Privacy Violation')
Medium	Serializable Class Containing Sensitive Data	OPT.OBJECTIVEC.SECURITY.SerializableClassContainingSensitiveData	SerializableClassContainingSensitiveData: Serializable Class Containing Sensitive Data
Medium	Third Party Keyboard Allowed	OPT.OBJECTIVEC.SECURITY.ThirdPartyKeyboardAllowed	ThirdPartyKeyboardAllowed: Avoid exposing sensitive data to third party keyboards.

Oracle Forms Scan rules

Contrast Scan supports these rules for Oracle Forms.

Severity	Engine rule ID	Contrast rule	Description
Critical	OPT.ORACLEFORMS.SqlInjection	SQL Injection	SqlInjection: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
Critical	OPT.ORACLEFORMS.AvoidGoto	Avoid Goto	AvoidGoto: Do not use GOTO statement
Critical	OPT.ORACLEFORMS.DeleteWithoutWhere	Delete Without Where	DSW: Find DELETE queries without WHERE
Critical	OPT.ORACLEFORMS.DoNotUseCallOpenForm	Do Not Use Call Open Form	DoNotUseCallOpenForm: Do not use CALL_FORM or OPEN_FORM built-in in Oracle Applications
Critical	OPT.ORACLEFORMS.GroupByWithFieldsNotInSelect	Group By With Fields Not In Select	TotalGroupAgr: Do not use GROUP BY in fields that are not present in the select
Critical	OPT.ORACLEFORMS.GroupByWithoutAggregationFunction	Group By Without Aggregation Function	TotalGroup: Do not use GROUP BY in selects without aggregation functions
Critical	OPT.ORACLEFORMS.QueriesAfterRaise	Queries After Raise	ESPR: Do not put queries after RAISE and RAISE_APPLICATION_ERROR
High	OPT.ORACLEFORMS.UseBindVariables	Use Bind Variables	UVB: Use BIND variables
High	OPT.ORACLEFORMS.AvoidJoinsOnTooManyTables	Avoid Joins On Too Many Tables	AvoidJoinsOnTooManyTables: Avoid joins between too many tables
High	OPT.ORACLEFORMS.AvoidMultipleOrInWhere	Avoid Multiple Or In Where	TooMuchOr: Do not perform several OR checks over the same field
High	OPT.ORACLEFORMS.AvoidNegatedWhereClauses	Avoid Negated Where Clauses	AvoidNeg: Do not use negations in the WHERE clauses
High	OPT.ORACLEFORMS.AvoidSelectAsterisk	Avoid Select Asterisk	ICT: Put columns of the table to query
High	OPT.ORACLEFORMS.AvoidSqlInTriggers	Avoid SQL In Triggers	AvoidSqlInTriggers: Avoid having SQL code in certain trigger types
High	OPT.ORACLEFORMS.AvoidStartingPercentInLike	Avoid Starting Percent In Like	AvoidPercent: Warns about queries that use LIKE filters and '%' patterns

Severity	Engine rule ID	Contrast rule	Description
High	OPT.ORACLEFORMS.AvoidTooLargePlsqlCode	Avoid Too Large Plsql Code	NLSM: Find PLSQL with more than 1000 lines
High	OPT.ORACLEFORMS.AvoidTooLargeRoutines	Avoid Too Large Routines	BigSize: Detects functions and procedures too large
High	OPT.ORACLEFORMS.CheckNoDataFound	Check No Data Found	NDFException: Check NO_DATA_FOUND exception when SELECT with INTO statement is used
High	OPT.ORACLEFORMS.CloseOpenedCursors	Close Opened Cursors	CC: Close all opened cursors
High	OPT.ORACLEFORMS.CloseOpenedRefCursors	Close Opened Ref Cursors	CRC: Close all opened ref cursors
High	OPT.ORACLEFORMS.DoNotCallHost	Do Not Call Host	DoNotCallHost: Do not call HOST built-in
High	OPT.ORACLEFORMS.DoNotRaiseApplicationError	Do Not Raise Application Error	DoNotRaiseApplicationError: Do not use RAISE_APPLICATION_ERROR
High	OPT.ORACLEFORMS.DoNotRepeatSqlCode	Do Not Repeat SQL Code	DoNotRepeatSqlCode: Do not copy and paste SQL code
High	OPT.ORACLEFORMS.DoNotUseGlobalVariables	Do Not Use Global Variables	DoNotUseGlobalVariables: Do not use Forms Global Variables
High	OPT.ORACLEFORMS.DoNotUseSqlControlVarsOutsideException	Do Not Use SQL Control Vars Outside Exception	OracleVar: Do not use control variables outside EXCEPTION blocks
High	OPT.ORACLEFORMS.ParametersByReferenceInCalls	Parameters By Reference In Calls	PPR: Use reference parameters
High	OPT.ORACLEFORMS.RoutineMustControlExceptions	Routine Must Control Exceptions	GER1: Must be at least a block of exceptions by routine
High	OPT.ORACLEFORMS.UpdateWithoutWhere	Update Without Where	USW: Find UPDATE queries without WHERE
High	OPT.ORACLEFORMS.UseInInsteadOfOr	Use In Instead Of Or	UILO: Use IN instead of OR
High	OPT.ORACLEFORMS.UseProperCoordinateSystem	Use Proper Coordinate System	UseProperCoordinateSystem: Use proper coordinate system
High	OPT.ORACLEFORMS.UseWhileInsteadOfExitWhen	Use While Instead Of Exit When	WL: Use WHILE instead of EXIT WHEN
High	OPT.ORACLEFORMS.WhenOthersInExceptionControl	When Others In Exception Control	GER2: WHEN OTHERS clause must be included in exceptions
Info	OPT.ORACLEFORMS.AliasInSelectFields	Alias In Select Fields	JOIN2: Use the specific alias for JOINS
Info	OPT.ORACLEFORMS.AvoidDatabaseLinks	Avoid Database Links	DBL: Find @dblink

Severity	Engine rule ID	Contrast rule	Description
Info	OPT.ORACLEFORMS.NamingConvention	Naming Convention	NamingConvention: Name of Oracle Forms/Reports elements must match naming conventions
Info	OPT.ORACLEFORMS.UseConsistentJoinSyntax	Use Consistent Join Syntax	DefSyntax: Defines the syntax in the SELECT statements
Info	OPT.ORACLEFORMS.UseTableAlias	Use Table Alias	TableAlias: Define an alias for each table
Low	OPT.ORACLEFORMS.AvoidSubqueries	Avoid Subqueries	InSelects: Do not use SELECT with subqueries in the FROM or WHERE clauses
Low	OPT.ORACLEFORMS.Comment	Comment	Comment: Avoid objects without comment property
Low	OPT.ORACLEFORMS.ElementNameMustEqualDatabaseName	Element Name Must Equal Database Name	ElementNameMustEqualDatabaseName: Blocks and Items should have the same table and column as the element name
Medium	OPT.ORACLEFORMS.AvoidForInCursors	Avoid For In Cursors	NDCF: Avoid declaring cursors 'on the fly'
Medium	OPT.ORACLEFORMS.AvoidNonBlockingInteractionMode	Avoid Non Blocking Interaction Mode	AvoidNonBlockingInteractionMode: Avoid Non-Blocking Interaction Mode in forms
Medium	OPT.ORACLEFORMS.AvoidUnusedLocalVars	Avoid Unused Local Vars	UselessVar: Detects local variables declared but not used
Medium	OPT.ORACLEFORMS.AvoidUsingDataDictionary	Avoid Using Data Dictionary	OracleTables: Avoid using tables and views of the Oracle Data Dictionary
Medium	OPT.ORACLEFORMS.CaseWithoutExcludingConditions	Case Without Excluding Conditions	UndefCase: Checks WHEN clauses are using the same control variable
Medium	OPT.ORACLEFORMS.DoNotReferenceFormObjectsInLibraries	Do Not Reference Form Objects In Libraries	DoNotReferenceFormObjectsInLibraries: Do not reference Form objects in libraries
Medium	OPT.ORACLEFORMS.FunctionsInWhere	Functions In Where	EFCW: Avoid use of functions in WHERE clause
Medium	OPT.ORACLEFORMS.UselessParam	Useless Param	UselessParam: Detects parameters declared but not used

PHP Scan rules

Contrast Scan supports these rules for PHP.

Severity	Contrast rule	Engine rule ID	Description
Critical	Insecure Php Configuration	OPT.PHP.InsecurePhpConfiguration	InsecurePhpConfiguration: Avoid insecure configuration settings in php.ini / .htaccess descriptors
Critical	Too Broad CORS Policy	OPT.PHP.TooBroadCORSPolicy	TooBroadCORSPolicy: CORS policy (Cross-origin resource sharing) too broad
Critical	Code Injection	OPT.PHP.CodeInjection	CodeInjection: Improper Neutralization of Directives in Dynamically Evaluated Code ('Eval Injection')

Severity	Contrast rule	Engine rule ID	Description
Critical	Command Injection	OPT.PHP.CommandInjection	CommandInjection: Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
Critical	Connection String Parameter Pollution	OPT.PHP.ConnectionStringParameterPollution	ConnectionStringParameterPollution: Connection string polluted with untrusted input
Critical	Cross Site Scripting	OPT.PHP.CrossSiteScripting	CrossSiteScripting: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
Critical	Csv Formula Injection	OPT.PHP.CsvFormulaInjection	CsvFormulaInjection: CSV Excel macro injection
Critical	DoS Regexp	OPT.PHP.DoSRegexp	DoSRegexp: Prevent denial of service attack through malicious regular expression ('Regex Injection')]
Critical	External Variable Modification	OPT.PHP.ExternalVariableModification	ExternalVariableModification: PHP External Variable Modification
Critical	Http Parameter Pollution	OPT.PHP.HttpParameterPollution	HttpParameterPollution: HTTP parameter pollution (HPP)
Critical	Http Splitting	OPT.PHP.HttpSplitting	HttpSplitting: Improper Neutralization of CRLF Sequences in HTTP Headers ('HTTP Response Splitting')
Critical	Ldap Injection	OPT.PHP.LdapInjection	LdapInjection: Avoid non-neutralized user-controlled input in LDAP search filters
Critical	Mail Header Manipulation	OPT.PHP.MailHeaderManipulation	MailHeaderManipulation: SMTP Header manipulation
Critical	Open Redirect	OPT.PHP.OpenRedirect	OpenRedirect: URL Redirection to Untrusted Site ('Open Redirect')
Critical	Resource Injection	OPT.PHP.ResourceInjection	ResourceInjection: Improper Control of Resource Identifiers ('Resource Injection')
Critical	Mail Command Injection	OPT.PHP.SEC.MailCommandInjection	MailCommandInjection: Mail Command Injection
Critical	No SQL Injection	OPT.PHP.SEC.NoSQLInjection	NoSQLInjection: Improper neutralization of special elements in data query logic (NoSQL injection)
Critical	Server Side Request Forgery	OPT.PHP.ServerSideRequestForgery	ServerSideRequestForgery: Server-Side Request Forgery (SSRF)
Critical	SQL Injection	OPT.PHP.SqlInjection	SqlInjection: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
Critical	Xml Entity Injection	OPT.PHP.XmlEntityInjection	XmlEntityInjection: XML entity injection
Critical	Avoid Use Default Secret	OPT.PHP.AvoidUseDefaultSecret	AvoidUseDefaultSecret: Avoid using secret value default symfony: ThisTokenIsNotSoSecretChangelt
Critical	Assign Null In Function Call	OPT.PHP.AssignNullInFunctionCall	AssignNullInFunctionCall: Assignment of a function call to a variable when the function does not have a return value
Critical	Avoid Exitor Die	OPT.PHP.AvoidExitorDie	AvoidExitorDie: Do not use exit() or die() for error processing
Critical	Avoid Global Variables within Functions	OPT.PHP.AvoidGlobalVariableswithinFunctions	AvoidGlobalVariableswithinFunctions: Avoid global variables within functions
Critical	Avoid Loop With Empty Body	OPT.PHP.AvoidLoopWithEmptyBody	AvoidLoopWithEmptyBody: Avoid loops (while, do/while, for) with empty body
Critical	Avoid SQL Queries Within Loop	OPT.PHP.AvoidSQLQueriesWithinLoop	AvoidSQLQueriesWithinLoop: Avoid doing SQL queries within a loop

Severity	Contrast rule	Engine rule ID	Description
Critical	Avoid This In Static Methods	OPT.PHP.AvoidThisInStaticMethods	AvoidThisInStaticMethods: Avoid \$this in static methods
Critical	Avoid Using Echo HTML	OPT.PHP.AvoidUsingEchoHTML	AvoidUsingEchoHTML: Avoid using echo or print to construct HTML
Critical	Dangerous File Upload	OPT.PHP.DangerousFileUpload	DangerousFileUpload: Unrestricted Upload of File with Dangerous Type
Critical	Do Not Use Error Suppression	OPT.PHP.DoNotUseErrorSuppression	DoNotUseErrorSuppression: Do not use error suppression with @
Critical	Fingers Crossed Logger In Production	OPT.PHP.FingersCrossedLoggerInProduction	FingersCrossedLoggerInProduction: Logging should not use many resources in production.
Critical	Function Arguments Uniqueness	OPT.PHP.FunctionArgumentsUniqueness	FunctionArgumentsUniqueness: Avoid duplicated argument names in function declarations
Critical	Include File Injection	OPT.PHP.IncludeFileInjection	IncludeFileInjection: Improper Control of filename for include / require statement
Critical	Nested If Statements	OPT.PHP.NestedIfStatements	NestedIfStatements: Avoid a high level of if statement nesting
Critical	Optional Parameters At End	OPT.PHP.OptionalParametersAtEnd	OptionalParametersAtEnd: Optional parameters in a function or method declaration must be always declared at the end
Critical	Path Traversal	OPT.PHP.PathTraversal	PathTraversal: Avoid non-neutralized user-controlled input to be part of a pathname (file or directory) used in I/O operations
Critical	Persistent Database Connections	OPT.PHP.PersistentDatabaseConnections	PersistentDatabaseConnections: Use persistent database connections
Critical	POSIX Extended Regular Expressions	OPT.PHP.POSIXExtendedRegularExpressions	POSIXExtendedRegularExpressions: Do not use POSIX Extended regular expression functions
Critical	Return In Constructor	OPT.PHP.ReturnInConstructor	ReturnInConstructor: Constructor returns a value
Critical	Too Many Parameters In Call	OPT.PHP.TooManyParametersInCall	TooManyParametersInCall: Avoid calling a function or method with more parameters than declared
Critical	Password In Redirect Rule	OPT.PHP.SEC.PasswordInRedirectRule	PasswordInRedirectRule: Password Management - Password in Redirect
Critical	Sensitive Data Non Parameter	OPT.PHP.SensitiveDataNonParameter	SensitiveDataNonParameter: Sensitive data such as database connection must be in a parameter file
High	Cake PHP Configuration	OPT.PHP.CakePHPConfiguration	CakePHPConfiguration: CakePHP framework weak configuration
High	Cookies Configuration	OPT.PHP.CookiesConfiguration	CookiesConfiguration: Weak cookies configuration
High	Insufficient Session Expiration Rule	OPT.PHP.SEC.InsufficientSessionExpirationRule	InsufficientSessionExpirationRule: Checks that session expiration interval does not exceed a limit
High	Session Cookie Configuration	OPT.PHP.SessionCookieConfiguration	SessionCookieConfiguration: Weak session cookies configuration
High	Zend Configuration	OPT.PHP.ZendConfiguration	ZendConfiguration: Zend framework session management configuration
High	Avoid Eval	OPT.PHP.AvoidEval	AvoidEval: Do not use eval()
High	Cross Site Request Forgery	OPT.PHP.CrossSiteRequestForgery	CrossSiteRequestForgery: Cross-Site Request Forgery (CSRF)

Severity	Contrast rule	Engine rule ID	Description
High	Enabled Twig Auto Escaping	OPT.PHP.EnabledTwigAutoEscaping	EnabledTwigAutoEscaping: Twig auto-escaping must be enabled
High	External Control Of Configuration Setting	OPT.PHP.SEC.ExternalControlOfConfigurationSetting	ExternalControlOfConfigurationSetting: External Control of System or Configuration Setting
High	JSON Injection	OPT.PHP.SEC.JSONInjection	JSONInjection: Avoid using non-neutralized user-controlled input into JSON entities - JSON Injection
High	Trust Boundary Violation Rule	OPT.PHP.SEC.TrustBoundaryViolationRule	TrustBoundaryViolationRule: Trust boundary violation
High	Xslt Injection	OPT.PHP.SEC.XsltInjection	XsltInjection: XML Injection (aka Blind XPath Injection)
High	Stored Cross Site Scripting	OPT.PHP.StoredCrossSiteScripting	StoredCrossSiteScripting: Improper neutralization of stored data during web content generation (Cross-site Scripting, XSS)
High	XPath Injection	OPT.PHP.XPathInjection	XPathInjection: Improper Neutralization of Data within XPath Expressions ('XPath Injection')
High	Assign Objects In Instantiation	OPT.PHP.AssignObjectsInInstantiation	AssignObjectsInInstantiation: Assign objects to a variable in instantiation
High	Avoid Business Logic In Twig	OPT.PHP.AvoidBusinessLogicInTwig	AvoidBusinessLogicInTwig: Avoid too much logic in Twig templates
High	Too Many Statements In Case	OPT.PHP.TooManyStatementsInCase	TooManyStatementsInCase: Avoid using too many statements in each case of a switch statement
High	Avoid Auto Reload In Twig	OPT.PHP.AvoidAutoReloadInTwig	AvoidAutoReloadInTwig: Twig auto_reload must be disabled
High	Avoid Create From Globals Request	OPT.PHP.AvoidCreateFromGlobalsRequest	AvoidCreateFromGlobalsRequest: Avoid use Request::createFromGlobals method
High	Avoid High Number Of Files In Folder	OPT.PHP.AvoidHighNumberOfFilesInFolder	AvoidHighNumberOfFilesInFolder: Avoid handling of folders with many files
High	Avoid Logical Operators	OPT.PHP.AvoidLogicalOperators	AvoidLogicalOperators: Avoid using logical operators
High	Avoid Ref With Multidim Array	OPT.PHP.AvoidRefWithMultidimArray	AvoidRefWithMultidimArray: Avoid &-ref-operator to substitute (or alias) a complex mutidim-array
High	Avoid Sleep Function	OPT.PHP.AvoidSleepFunction	AvoidSleepFunction: Avoid using the sleep function
High	Avoid Strict Variables In Twig	OPT.PHP.AvoidStrictVariablesInTwig	AvoidStrictVariablesInTwig: Twig strict_variables must be disabled
High	Avoid Action Method Too Long	OPT.PHP.AvoidActionMethodTooLong	AvoidActionMethodTooLong: Avoid using methods controllers "action" too long
High	Break Non Empty Switch Clauses	OPT.PHP.BreakNonEmptySwitchClauses	BreakNonEmptySwitchClauses: Use break statement at the last statement of SwitchCase
High	Call Time Pass By Reference Forbidden	OPT.PHP.CallTimePassByReferenceForbidden	CallTimePassByReferenceForbidden: Do not use call-time pass-by-reference arguments in function calls
High	Catching Exception	OPT.PHP.CatchingException	CatchingException: Avoid catching exception base classes

Severity	Contrast rule	Engine rule ID	Description
High	Check Parameters Number In Function	OPT.PHP.CheckParametersNumberInFunction	CheckParametersNumberInFunction: Too many parameters in function
High	Class Includes	OPT.PHP.ClassIncludes	ClassIncludes: include / require / require_once / include_once is not allowed for loading class files
High	Delete Acme Demo Bundle	OPT.PHP.DeleteAcmeDemoBundle	DeleteAcmeDemoBundle: Perform deletion of the AcmeDemoBundle that is included by default in Symfony2
High	Delete Unused Libraries	OPT.PHP.DeleteUnusedLibraries	DeleteUnusedLibraries: Remove references to libraries that are not used
High	Default Arguments On The Right Side	OPT.PHP.DefaultArgumentsOnTheRightSide	DefaultArgumentsOnTheRightSide: Default arguments must be on the right side of any non-default arguments
High	Default Clause Switch Statements	OPT.PHP.DefaultClauseSwitchStatements	DefaultClauseSwitchStatements: Use default clause at the end of the switch statement
High	Efficient Php Ini Configuration	OPT.PHP.EfficientPhpIniConfiguration	EfficientPhpIniConfiguration: Certain configuration properties should be set for efficiency
High	Establish New Path Cache	OPT.PHP.EstablishNewPathCache	EstablishNewPathCache: Set a new path for the cache
High	Fav Icon In Web Directory	OPT.PHP.FaviconInWebDirectory	FaviconInWebDirectory: Use favicon in the web application
High	Few Action Methods In Controller	OPT.PHP.FewActionMethodsInController	FewActionMethodsInController: Use a controller by specific concept
High	Get Action Should Not Modify Resources	OPT.PHP.GetActionShouldNotModifyResources	GetActionShouldNotModifyResources: Use GET method only to get information
High	Include Require Without Parentheses	OPT.PHP.IncludeRequireWithoutParentheses	IncludeRequireWithoutParentheses: Do not use include and its variants with parentheses
High	Keywords Case	OPT.PHP.KeywordsCase	KeywordsCase: Write PHP keywords in lower-case
High	Many Cases	OPT.PHP.ManyCases	ManyCases: Avoid too many choices in switch structures
High	Max Methods	OPT.PHP.MaxMethods	MaxMethods: Maximum allowed number of methods
High	Missing Authorization	OPT.PHP.MissingAuthorization	MissingAuthorization: Inadequate authorization check to access a resource or perform an action
High	No Update Loop Vars In For Body	OPT.PHP.NoUpdateLoopVarsInForBody	NoUpdateLoopVarsInForBody: Do not update control vars in 'for' loop body
High	No Use Flush In Loop	OPT.PHP.NoUseFlushInLoop	NoUseFlushInLoop: Avoid calling to the flush() method within a loop
High	No Use Data Base Functions Specific Provider	OPT.PHP.NoUseDataBaseFunctionsSpecificProvider	NoUseDataBaseFunctionsSpecificProvider: Avoid use specific functions of a database provider
High	No Use PHP Response Functions	OPT.PHP.NoUsePHPResponseFunctions	NoUsePHPResponseFunctions: Avoid using php response functions
High	No Use PHP Session Functions	OPT.PHP.NoUsePHPSessionFunctions	NoUsePHPSessionFunctions: Do not use PHP session functions

Severity	Contrast rule	Engine rule ID	Description
High	No Use PHP Super Global	OPT.PHP.NoUsePHPSuperGlobal	NoUsePHPSuperGlobal: Use Request Object instead of super-global variables PHP
High	Numerically Indexed Arrays	OPT.PHP.NumericallyIndexedArrays	NumericallyIndexedArrays: Negative numbers are not permitted as indices
High	Php Tags	OPT.PHP.PhpTags	PhpTags: Do not use the short form for the PHP opening tag
High	Public Method Only Actions	OPT.PHP.PublicMethodOnlyActions	PublicMethodOnlyActions: In controllers can only be public the "Action" methods
High	Rethrowing Exceptions	OPT.PHP.RethrowingExceptions	RethrowingExceptions: When rethrowing, the original exception must be wrapped inside the one being thrown
High	Return Value Ignored	OPT.PHP.ReturnValueIgnored	ReturnValueIgnored: Function call return value ignored
High	Robots Txt In Web Directory	OPT.PHP.RobotsTxtInWebDirectory	RobotsTxtInWebDirectory: Use robots.txt in the web application
High	Routes Should Reference Existing Actions	OPT.PHP.RoutesShouldReferenceExistingActions	RoutesShouldReferenceExistingActions: Avoid routes that reference a non-existent actions
High	Cookies In Security Decision	OPT.PHP.SEC.CookiesInSecurityDecision	CookiesInSecurityDecision: Reliance on Cookies without Validation and Integrity Checking in a Security Decision
High	User Controlled SQL Primary Key	OPT.PHP.SEC.UserControlledSQLPrimaryKey	UserControlledSQLPrimaryKey: Avoid using an user controlled Primary Key into a query.
High	Should Not Throw Access Denied Http Exception	OPT.PHP.ShouldNotThrowAccessDeniedHttpException	ShouldNotThrowAccessDeniedHttpException: Avoid throwing AccessDeniedHttpException
High	Smart Substring Matching	OPT.PHP.SmartSubstringMatching	SmartSubstringMatching: Smart substring matching
High	Too Many Break Or Continue In Loop	OPT.PHP.TooManyBreakOrContinueInLoop	TooManyBreakOrContinueInLoop: Avoid using more than one break or continue statement in each loop
High	Throwing Exception	OPT.PHP.ThrowingException	ThrowingException: Avoid throwing exception base classes
High	Unused Function Parameter	OPT.PHP.UnusedFunctionParameter	UnusedFunctionParameter: Avoid unused function parameters
High	Unused Local Var	OPT.PHP.UnusedLocalVar	UnusedLocalVar: Avoid unused local variables
High	Unused Private Field	OPT.PHP.UnusedPrivateField	UnusedPrivateField: Avoid unused private fields
High	Unused Private Method	OPT.PHP.UnusedPrivateMethod	UnusedPrivateMethod: Avoid unused private methods
High	Use Redirect After Posting Data	OPT.PHP.UseRedirectAfterPostingData	UseRedirectAfterPostingData: Is recommended to use redirect after doing a POST
High	Avoid Inject Request Service	OPT.PHP.AvoidInjectRequestService	AvoidInjectRequestService: Avoid injecting the service request
High	Do Not Debug In Twig Templates	OPT.PHP.DoNotDebugInTwigTemplates	DoNotDebugInTwigTemplates: Avoid using debug tag in Twig templates
High	Http To Send Data	OPT.PHP.HttpToSendData	HttpToSendData: Avoid using HTTP instead of HTTPS

Severity	Contrast rule	Engine rule ID	Description
High	Information Exposure Through Error Message	OPT.PHP.InformationExposureThroughErrorMessage	InformationExposureThroughErrorMessage : Avoid sensitive information exposure through error messages
High	Password Management	OPT.PHP.PasswordManagement	PasswordManagement: Use of empty or hardcoded password, or storing password in comments
High	Hardcoded Crypto Key	OPT.PHP.HardcodedCryptoKey	HardcodedCryptoKey: Use of Hard-coded Cryptographic Key
High	Hardcoded Salt	OPT.PHP.HardcodedSalt	HardcodedSalt: Use of hardcoded salt
High	Insecure Randomness	OPT.PHP.InsecureRandomness	InsecureRandomness: Standard pseudo-random number generators cannot withstand cryptographic attacks
High	Missing Encryption Of Sensitive Data	OPT.PHP.MissingEncryptionOfSensitiveData	MissingEncryptionOfSensitiveData: Encrypt sensitive data before transmission or storage
High	Insufficient Key Size Rule	OPT.PHP.SEC.InsufficientKeySizeRule	InsufficientKeySizeRule: Weak cryptography, insufficient key length
High	Weak Cryptographic Hash	OPT.PHP.WeakCryptographicHash	WeakCryptographicHash: Weak cryptographic hash
High	Weak Encryption Algorithm	OPT.PHP.WeakEncryptionAlgorithm	WeakEncryptionAlgorithm: Weak symmetric encryption algorithm
Info	Log Forging	OPT.PHP.LogForging	LogForging: Improper Output Neutralization for Logs
Info	Change Default Favicon	OPT.PHP.ChangeDefaultFavicon	ChangeDefaultFavicon: Replace the default favicon.ico icon
Info	Line Length	OPT.PHP.LineLength	LineLength: Lines should not exceed a maximum size
Info	Methods Name Convention	OPT.PHP.MethodsNameConvention	MethodsNameConvention: Function / method names should comply with the naming convention
Info	Php Comments Rule	OPT.PHP.PhpCommentsRule	PhpCommentsRule: Code must be commented according to comments standard
Info	Variable Substitution	OPT.PHP.VariableSubstitution	VariableSubstitution: Avoid using \${...} style for variable substitutions
Info	Avoid Using Request	OPT.PHP.AvoidUsingRequest	AvoidUsingRequest: Using \$_REQUEST is strongly discouraged
Low	Do Not Use Default Session Cookies Name	OPT.PHP.DoNotUseDefaultSessionCookiesName	DoNotUseDefaultSessionCookiesName: Do not use the default session cookie's name
Low	Avoid Complex Methods	OPT.PHP.AvoidComplexMethods	AvoidComplexMethods: Avoid too long methods
Low	Avoid Functions In Loops	OPT.PHP.AvoidFunctionsInLoops	AvoidFunctionsInLoops: Avoid calling functions from evaluation or update part of the loops
Low	Avoid Many Variables In Twig	OPT.PHP.AvoidManyVariablesInTwig	AvoidManyVariablesInTwig: Avoid return too many variables to a Twig template
Low	Avoid Special Comment	OPT.PHP.AvoidSpecialComment	AvoidSpecialComment: Avoid using comments to indicate parts of code that need to be revised
Low	Avoid Too Long Twig Templates	OPT.PHP.AvoidTooLongTwigTemplates	AvoidTooLongTwigTemplates: Too long Twig templates make code harder to understand and maintain

Severity	Contrast rule	Engine rule ID	Description
Low	Avoid Var Name Prefix Is	OPT.PHP.AvoidVarNamePrefixIs	AvoidVarNamePrefixIs: PHP variables should not have the prefix "is"
Low	Class Format PSR-0	OPT.PHP.ClassFormatPSR0	ClassFormatPSR0: PHP classes must have name and namespace compliant with PSR-0 (autoloading) standard
Low	Closing Php Tag	OPT.PHP.ClosingPhpTag	ClosingPhpTag: Omit closing tag in files with only php code
Low	Constants Name Convention	OPT.PHP.ConstantsNameConvention	ConstantsNameConvention: Constant names should comply with the naming convention
Low	Customize Error Pages	OPT.PHP.CustomizeErrorPages	CustomizeErrorPages: Customize Symfony error pages
Low	Declarations Without Side Effects	OPT.PHP.DeclarationsWithoutSideEffects	DeclarationsWithoutSideEffects: PHP files declaring global symbols (functions, classes) should not have side effects
Low	Dependency Injection Container	OPT.PHP.DependencyInjectionContainer	DependencyInjectionContainer: Hide dependencies between services
Low	Else In Else If Statement	OPT.PHP.ElseInElseIfStatement	ElseInElseIfStatement: Else if statements should finish with an else clause
Low	Empty Boot Method In Bundle	OPT.PHP.EmptyBootMethodInBundle	EmptyBootMethodInBundle: The boot() method of all bundles must be empty
Low	Exception Extension	OPT.PHP.ExceptionExtension	ExceptionExtension: Properly custom classes that extend from Exception class
Low	Modifiers Order	OPT.PHP.ModifiersOrder	ModifiersOrder: Abstract, final and static modifiers declaration order
Low	Names Suffix Conventions	OPT.PHP.NamesSuffixConventions	NamesSuffixConventions: Comply with the naming conventions for interfaces, traits and exception classes
Low	Not Mix End Of Lines	OPT.PHP.NotMixEndOfLines	NotMixEndOfLines: The files must have a only newline format
Low	No Use Debug Statements	OPT.PHP.NoUseDebugStatements	NoUseDebugStatements: No use debug statement in production
Low	Organize Forms In Categories	OPT.PHP.OrganizeFormsInCategories	OrganizeFormsInCategories: Forms directory must be structured
Low	Improper Validation Of Array Index	OPT.PHP.SEC.ImproperValidationOfArrayIndex	ImproperValidationOfArrayIndex: Array index coming from a non neutralized vulnerable input
Low	Specify The Allowed Methods For The Routes	OPT.PHP.SpecifyTheAllowedMethodsForTheRoutes	SpecifyTheAllowedMethodsForTheRoutes: Specify the allowed methods for each route
Low	String Concatenation	OPT.PHP.StringConcatenation	StringConcatenation: Add an space before and after '.' operator. In multiple line concatenating statements, pad each line to align '.' operator under ' {}
Low	Text Files End Properly	OPT.PHP.TextFilesEndProperly	TextFilesEndProperly: The files must end with newline character
Low	Type Hint Object Arguments	OPT.PHP.TypeHintObjectArguments	TypeHintObjectArguments: Specify argument's type of each method
Low	Use Strict Comparisons	OPT.PHP.UseStrictComparisons	UseStrictComparisons: Use strict comparisons
Low	Variable As String	OPT.PHP.VariableAsString	VariableAsString: Use of a string created just from a variable
Medium	Avoid Contain Config File	OPT.PHP.AvoidContainConfigFile	AvoidContainConfigFile: config.php should be removed before the project goes into production

Severity	Contrast rule	Engine rule ID	Description
Medium	Plaintext Storage In A Cookie Rule	OPT.PHP.SEC.PlaintextStorageInACookieRule	PlaintextStorageInACookieRule: Cleartext Storage of Sensitive Information in a Cookie
Medium	Cross Site History Manipulation	OPT.PHP.SEC.CrossSiteHistoryManipulation	CrossSiteHistoryManipulation: Cross-Site History Manipulation (XSHM)
Medium	Format String Injection Rule	OPT.PHP.SEC.FormatStringInjectionRule	FormatStringInjectionRule: Exclude unsanitized user input from format strings
Medium	Serialization Injection	OPT.PHP.SerializationInjection	SerializationInjection: Deserialization of untrusted data
Medium	Avoid Concat In Echo	OPT.PHP.AvoidConcatInEcho	AvoidConcatInEcho: Avoid concatenations in echo statement
Medium	Use Switch Instead Of If Else If	OPT.PHP.UseSwitchInsteadOfIfElseIf	UseSwitchInsteadOfIfElseIf: Avoid using if-elseif-else chains
Medium	Avoid Large Classes	OPT.PHP.AvoidLargeClasses	AvoidLargeClasses: Avoid classes with too many lines of code
Medium	Avoid Large Methods	OPT.PHP.AvoidLargeMethods	AvoidLargeMethods: Avoid functions and methods with too many lines of code
Medium	Avoid Magic Numbers	OPT.PHP.AvoidMagicNumbers	AvoidMagicNumbers: Avoid literals in method calls. Named constants make source code easier to understand and maintain
Medium	Avoid Pass Entity Manager As Argument	OPT.PHP.AvoidPassEntityManagerAsArgument	AvoidPassEntityManagerAsArgument: Avoid passing as arguments the entity manager
Medium	Avoid Unnecessary Replacements In Loops	OPT.PHP.AvoidUnnecessaryReplacementsInLoops	AvoidUnnecessaryReplacementsInLoops: Avoid unnecessary string replacements in loops
Medium	Avoid Unreachable Code	OPT.PHP.AvoidUnreachableCode	AvoidUnreachableCode: Avoid implementing code will never be executed
Medium	Check Field Number In Class	OPT.PHP.CheckFieldNumberInClass	CheckFieldNumberInClass: Avoid classes with too many fields by class
Medium	Check Public Methods Number In Class	OPT.PHP.CheckPublicMethodsNumberInClass	CheckPublicMethodsNumberInClass: Avoid classes with too many methods
Medium	Close Opened Database Connections	OPT.PHP.CloseOpenedDatabaseConnections	CloseOpenedDatabaseConnections: Close non-persistent connections as soon as they are no longer required
Medium	Counter Functions In Loops	OPT.PHP.CounterFunctionsInLoops	CounterFunctionsInLoops: Do not use counter functions in loop expressions
Medium	Exceptions Disable In Production	OPT.PHP.ExceptionsDisableInProduction	ExceptionsDisableInProduction: Disable exceptions when the application is in production
Medium	Foreach To Loop Through Arrays	OPT.PHP.ForeachToLoopThroughArrays	ForeachToLoopThroughArrays: Use foreach to loop through arrays
Medium	If Variable To Check Initialization	OPT.PHP.IfVariableToCheckInitialization	IfVariableToCheckInitialization: Do not use if (\$var) to check if a variable is initialized
Medium	MIME Type Detection	OPT.PHP.MIMETypeDetection	MIMETypeDetection: Avoid MIME type detection using 'type' attribute of a file in \$_FILES array
Medium	No Use Deprecated Functions	OPT.PHP.NoUseDeprecatedFunctions	NoUseDeprecatedFunctions: Do not use deprecated functions

Severity	Contrast rule	Engine rule ID	Description
Medium	No Use Magic Constant __DIR__ and __FILE__	OPT.PHP.NoUseMagicConstant__DIR__and__FILE__ —	NoUseMagicConstant__DIR__and__FILE__: Avoid using of the magic constants (__FILE__ and __DIR__)
Medium	Num Max Class By Namespace	OPT.PHP.NumMaxClassByNamespace	NumMaxClassByNamespace: Avoid an excessive number of classes per package/namespace
Medium	Return Value Without Parentheses	OPT.PHP.ReturnValueWithoutParentheses	ReturnValueWithoutParentheses: The return value must not be enclosed in parentheses
Medium	Execution After Redirect	OPT.PHP.SEC.ExecutionAfterRedirect	ExecutionAfterRedirect: Execution After Redirect (EAR)
Medium	Potential Infinite Loop	OPT.PHP.SEC.PotentialInfiniteLoop	PotentialInfiniteLoop: Loop with Unreachable Exit Condition ('Infinite Loop')
Medium	Unchecked Input In Loop Condition	OPT.PHP.SEC.UncheckedInputInLoopCondition	UncheckedInputInLoopCondition: Unchecked input in loop condition
Medium	Set For Attributes With Get	OPT.PHP.SetForAttributesWithGet	SetForAttributesWithGet: Create '__set' method when exists '__get' method
Medium	Single Line Comments	OPT.PHP.SingleLineComments	SingleLineComments: Do not use inline comments starting with the sharp character
Medium	String Literals	OPT.PHP.StringLiterals	StringLiterals: Use simple quotes to demarcate string literals, except when containing apostrophes, escaped chars or variable substitutions
Medium	Unsafe Function	OPT.PHP.UnsafeFunction	UnsafeFunction: Use of Potentially Dangerous Function
Medium	Use Latest Symfony Version	OPT.PHP.UseLatestSymfonyVersion	UseLatestSymfonyVersion: Use latest stable Symfony version
Medium	Use Maintained Symfony Version	OPT.PHP.UseMaintainedSymfonyVersion	UseMaintainedSymfonyVersion: Use maintained Symfony versions
Medium	Use Relative Path	OPT.PHP.UseRelativePath	UseRelativePath: Using relative paths to access resources
Medium	Variable Initialization	OPT.PHP.VariableInitialization	VariableInitialization: Variable initialization
Medium	Privacy Violation	OPT.PHP.PrivacyViolation	PrivacyViolation: Exposure of Private Information
Medium	Autocomplete On For Sensitive Fields	OPT.PHP.SEC.AutocompleteOnForSensitiveFields	AutocompleteOnForSensitiveFields: Autocomplete enabled for sensitive form fields
Medium	Plaintext Storage Of Password	OPT.PHP.SEC.PlaintextStorageOfPassword	PlaintextStorageOfPassword: Plaintext Storage of a Password

PL/SQL Scan rules

Contrast Scan supports these rules for PL/SQL.

Severity	Contrast rule	Engine rule ID	Description
Critical	Command Injection	OPT.PLSQL.SEC.CommandInjection	CommandInjection: Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')

Severity	Contrast rule	Engine rule ID	Description
Critical	Cross Site Scripting	OPT.PLSQL.SEC.CrossSiteScripting	CrossSiteScripting: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
Critical	Header Manipulation	OPT.PLSQL.SEC.HeaderManipulation	HeaderManipulation: Unvalidated data in HTTP response header or in cookies ('HTTP Response Splitting')
Critical	Persisted Cross Site Scripting	OPT.PLSQL.SEC.PersistedCrossSiteScripting	PersistedCrossSiteScripting: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
Critical	Second Order SQL Injection	OPT.PLSQL.SEC.SecondOrderSqlInjection	SecondOrderSqlInjection: SQL Injection (Second-Order).
Critical	Server Side Request Forgery	OPT.PLSQL.SEC.ServerSideRequestForgery	ServerSideRequestForgery: Server-Side Request Forgery (SSRF)
Critical	Sleep Injection	OPT.PLSQL.SEC.SleepInjection	SleepInjection: Denial of Service by externally controlled sleep time
Critical	SQL Injection	OPT.PLSQL.SEC.SqlInjection	SqlInjection: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
Critical	D V D P	OPT.PLSQL.DC_PLSQL.DVDP	DVDP: Do not declare variables out of the main Declare
Critical	N L S M	OPT.PLSQL.DC_PLSQL.NLSM	NLSM: Find PLSQL with more than 1000 lines
Critical	Pkg Comment	OPT.PLSQL.DOC_PLSQL.PkgComment	PkgComment: Packages without comments
Critical	C N L	OPT.PLSQL.GEN_PLSQL.CNL	CNL: CLOB, NO LONG
Critical	D E W O	OPT.PLSQL.GEN_PLSQL.DEWO	DEWO: Use 'RAISE_APPLICATION_ERROR' after 'EXCEPTION WHEN OTHERS'
Critical	D S W	OPT.PLSQL.GEN_PLSQL.DSW	DSW: Find DELETE queries without WHERE
Critical	E S P R	OPT.PLSQL.GEN_PLSQL.ESPR	ESPR: Do not put queries after RAISE and RAISE_APPLICATION_ERROR
Critical	I N C I	OPT.PLSQL.GEN_PLSQL.INCI	INCI: Put names to the columns in which data is inserted
Critical	J O I N	OPT.PLSQL.GEN_PLSQL.JOIN	JOIN: JOINS must have alias
Critical	T G	OPT.PLSQL.GEN_PLSQL.TG	TG: Identifying the existence of triggers to prevent problems of long-term maintainability
Critical	Transaction	OPT.PLSQL.GEN_PLSQL.Transaction	Transaction: Use of INSERT, UPDATE and DELETE with COMMIT and ROLLBACK invocations
Critical	U S W	OPT.PLSQL.GEN_PLSQL.USW	USW: Find UPDATE queries without WHERE
Critical	V A R 2	OPT.PLSQL.GEN_PLSQL.VAR2	VAR2: Define variables as VARCHAR2, nor as VARCHAR
Critical	Avoid Goto	OPT.PLSQL.MISC_PLSQL.AvoidGoto	AvoidGoto: Do not use GOTO statement
Critical	Dead Goto	OPT.PLSQL.MISC_PLSQL.DeadGoto	DeadGoto: Detects dead code after GOTO statements
Critical	Avoid calling LN	OPT.PLSQL.MISC_PLSQL.NLN	NLN: Avoid calling LN
Critical	P C D L	OPT.PLSQL.MISC_PLSQL.PCDL	PCDL: Link parameters supported by DatabaseLink
Critical	P D D	OPT.PLSQL.MISC_PLSQL.PDD	PDD: Passing parameters not associated to the DD in the Data Access Logic (LD) or Business Logic (LN)

Severity	Contrast rule	Engine rule ID	Description
Critical	P E P P	OPT.PLSQL.MISC_PLSQL.PEPP	PEPP: External packages must contain public procedures
Critical	P P A D	OPT.PLSQL.MISC_PLSQL.PPAD	PPAD: Passing parameters not associated to the DD in the Data Access Logic (LD)
Critical	Big Size	OPT.PLSQL.OYR_PLSQL.BigSize	BigSize: Detects functions and procedures too large
Critical	Check Data	OPT.PLSQL.OYR_PLSQL.CheckData	CheckData: It is not allowed to confirm the existence of a record before updating it
Critical	Dml Returning	OPT.PLSQL.OYR_PLSQL.DmlReturning	DmlReturning: Do not access to inserted, updated or deleted registers to obtain fields (before or after the operation)
Critical	E A D	OPT.PLSQL.OYR_PLSQL.EAD	EAD: Avoid DUAL access
Critical	Total Group	OPT.PLSQL.OYR_PLSQL.TotalGroup	TotalGroup: Do not use GROUP BY in selects without aggregation functions
Critical	Total Group Agr	OPT.PLSQL.OYR_PLSQL.TotalGroupAgr	TotalGroupAgr: Do not use GROUP BY in fields that are not present in the select
Critical	U C R	OPT.PLSQL.OYR_PLSQL.UCR	UCR: One process has a unique commit or rollback associated
Critical	Forbidden Call	OPT.PLSQL.SEC.ForbiddenCall	ForbiddenCall: Dangerous procedure / function called.
Critical	Path Traversal	OPT.PLSQL.SEC.PathTraversal	PathTraversal: External Control of File Name or Path
Critical	Too Broad Grant	OPT.PLSQL.SEC.TooBroadGrant	TooBroadGrant: Too broad privileges granted.
Critical	Weak Cryptographic Hash	OPT.PLSQL.SEC.WeakCryptographicHash	WeakCryptographicHash: Weak cryptographic hashes cannot guarantee data integrity
Critical	Weak Symmetric Encryption Algorithm	OPT.PLSQL.SEC.WeakSymmetricEncryptionAlgorithm	WeakSymmetricEncryptionAlgorithm: Weak symmetric encryption algorithm.
High	U V B	OPT.PLSQL.GEN_PLSQL.UVB	UVB: Use BIND variables
High	Open Redirect	OPT.PLSQL.SEC.OpenRedirect	OpenRedirect: Do not allow to control the URL used in a redirect by an unvalidated input
High	E C U	OPT.PLSQL.CNU_PLSQL.ECU	ECU: Avoid non-used constants
High	E V U	OPT.PLSQL.CNU_PLSQL.EVU	EVU: Avoid non-used variables
High	Useless Param	OPT.PLSQL.CNU_PLSQL.UselessParam	UselessParam: Detects parameters declared but not used
High	N D C F	OPT.PLSQL.DC_PLSQL.NDCF	NDCF: Avoid declaring cursors 'on the fly'
High	Proc Comment	OPT.PLSQL.DOC_PLSQL.ProcComment	ProcComment: Functions and procedures without comments
High	A M	OPT.PLSQL.GEN_PLSQL.AM	AM: Put a date mask to TO_DATE and TO_CHAR functions
High	Avoid Dual	OPT.PLSQL.GEN_PLSQL.AvoidDual	AvoidDual: Do not use SELECT over DUAL table
High	Avoid Inner	OPT.PLSQL.GEN_PLSQL.AvoidInner	AvoidInner: Detects the use of INNER JOIN
High	C O F	OPT.PLSQL.GEN_PLSQL.COF	COF: Do not use conditions and filter operators
High	D H	OPT.PLSQL.GEN_PLSQL.DH	DH: Avoid hints
High	G E R 2	OPT.PLSQL.GEN_PLSQL.GER2	GER2: WHEN OTHERS clause must be included in exceptions
High	G E R 3	OPT.PLSQL.GEN_PLSQL.GER3	GER3: Avoid WHEN OTHERS THEN NULL in exceptions

Severity	Contrast rule	Engine rule ID	Description
High	N D F Exception	OPT.PLSQL.GEN_PLSQL.NDFException	NDFException: Check NO_DATA_FOUND exception when SELECT with INTO statement is used
High	W L	OPT.PLSQL.GEN_PLSQL.WL	WL: Use WHILE instead of EXIT WHEN
High	E P P C	OPT.PLSQL.MISC_PLSQL.EPPC	EPPC: Avoid passing complex parameters
High	L E P	OPT.PLSQL.MISC_PLSQL.LEP	LEP: No parameters in external logic
High	Repeated Code	OPT.PLSQL.MISC_PLSQL.RepeatedCode	RepeatedCode: Avoid repeating the same query code
High	Avoid Func	OPT.PLSQL.OYR_PLSQL.AvoidFunc	AvoidFunc: Checks that there are no queries that use functions over fields of the tables in the WHERE condition
High	Count Asterisk	OPT.PLSQL.OYR_PLSQL.CountAsterisk	CountAsterisk: Do not use COUNT(*) function
High	Double Select	OPT.PLSQL.OYR_PLSQL.DoubleSelect	DoubleSelect: Do not use the same WHERE clause in two consecutive queries
High	Cursor Snarfing	OPT.PLSQL.SEC.CursorSnarfing	CursorSnarfing: Cursor Snarfing
High	Insecure Randomness	OPT.PLSQL.SEC.InsecureRandomness	InsecureRandomness: Standard pseudo-random number generators cannot withstand cryptographic attacks.
Info	Useless Query	OPT.PLSQL.CNU_PLSQL.UselessQuery	UselessQuery: It warns about queries that obtain the field which is used as filter in the WHERE clause
Info	C A B	OPT.PLSQL.DOC_PLSQL.CAB	CAB: Include comments in headers
Info	E N E C	OPT.PLSQL.FM_PLSQL.ENEC	ENEC: Names must not be between inverted commas
Info	L B	OPT.PLSQL.FM_PLSQL.LB	LB: Blank lines in some sentences
Info	M E I	OPT.PLSQL.FM_PLSQL.MEI	MEI: Code lines indentation
Info	Nomenclator	OPT.PLSQL.FM_PLSQL.Nomenclator	Nomenclator: Use a prefix in the name of variables, parameteres, exceptions and cursors
Info	DB L	OPT.PLSQL.GEN_PLSQL.DBL	DBL: Find @dblink
Info	Avoid independent procedures called from system logic	OPT.PLSQL.MISC_PLSQL.EPILS	EPILS: Avoid using independent procedures called from the system logic
Info	Avoid using procedures/ functions from DD	OPT.PLSQL.MISC_PLSQL.NPFD	NPFD: Avoid using procedures/ functions from DD
Info	N E D V	OPT.PLSQL.NOM_PLSQL.NEDV	NEDV: Specific naming convention for variables
Info	P A Q2	OPT.PLSQL.NOM_PLSQL.PAQ2	PAQ2: Specific naming convention for packages
Info	P R M A	OPT.PLSQL.NOM_PLSQL.PRMA	PRMA: Write reserved words in capital letters
Info	Condition Order	OPT.PLSQL.OYR_PLSQL.ConditionOrder	ConditionOrder: Looks for control contructions with more than one condition
Info	Default Authid	OPT.PLSQL.SEC.DefaultAuthid	DefaultAuthid: No explicit AUTHID clause.
Low	Useless Var	OPT.PLSQL.CNU_PLSQL.UselessVar	UselessVar: Detects local variables declared but not used

Severity	Contrast rule	Engine rule ID	Description
Low	Func Not Null	OPT.PLSQL.DC_PLSQL.FuncNotNull	FuncNotNull: Do not define parameters as NOT NULL in functions and procedures
Low	L C N I	OPT.PLSQL.DC_PLSQL.LCNI	LCNI: Incorrect literal and numeric constants
Low	Cap Word	OPT.PLSQL.FM_PLSQL.CapWord	CapWord: Use uppercase for PLSQL keywords
Low	C C	OPT.PLSQL.GEN_PLSQL.CC	CC: Close all opened cursors
Low	C R C	OPT.PLSQL.GEN_PLSQL.CRC	CRC: Close all opened ref cursors
Low	Do not qualify related tables	OPT.PLSQL.GEN_PLSQL.CTI	CTI: Do not qualify related tables
Low	G E R0	OPT.PLSQL.GEN_PLSQL.GER0	GER0: Make correct errors management
Low	G E R1	OPT.PLSQL.GEN_PLSQL.GER1	GER1: Must be at least a block of exceptions by routine
Low	J O I N2	OPT.PLSQL.GEN_PLSQL.JOIN2	JOIN2: Use the specific alias for JOINS
Low	N U L L	OPT.PLSQL.GEN_PLSQL.NULL	NULL: Use NULL instead of
Low	Oracle Tables	OPT.PLSQL.GEN_PLSQL.OracleTables	OracleTables: Avoid using tables and views of the Oracle Data Dictionary
Low	U I L O	OPT.PLSQL.GEN_PLSQL.UILO	UILO: Use IN instead of OR
Low	Use Constants	OPT.PLSQL.GEN_PLSQL.UseConstants	UseConstants: Do not use literals in the WHERE clauses
Low	Var Loop	OPT.PLSQL.GEN_PLSQL.VarLoop	VarLoop: Do not initialize variables in a loop
Low	C T E	OPT.PLSQL.NOM_PLSQL.CTE	CTE: Constant names must be written in capital letters
Low	Avoid Neg	OPT.PLSQL.OYR_PLSQL.AvoidNeg	AvoidNeg: Do not use negations in the WHERE clauses
Low	In Selects	OPT.PLSQL.OYR_PLSQL.InSelects	InSelects: Do not use SELECT with subqueries in the FROM or WHERE clauses
Low	N V L	OPT.PLSQL.OYR_PLSQL.NVL	NVL: Avoid using NVL function
Low	Use Between	OPT.PLSQL.OYR_PLSQL.UseBetween	UseBetween: Do not use '< [] {}'
Medium	C I F A	OPT.PLSQL.GEN_PLSQL.CIFA	CIFA: Use CASE instead nested if elseif
Medium	Def Syntax	OPT.PLSQL.GEN_PLSQL.DefSyntax	DefSyntax: Defines the syntax in the SELECT statements
Medium	E F C W	OPT.PLSQL.GEN_PLSQL.EFCW	EFCW: Avoid use of functions in WHERE clause
Medium	N A P E	OPT.PLSQL.GEN_PLSQL.NAPE	NAPE: Do not access to the to the errors stack
Medium	N F S0	OPT.PLSQL.GEN_PLSQL.NFS0	NFS0: Do not format output
Medium	N F S1	OPT.PLSQL.GEN_PLSQL.NFS1	NFS1: Do not format the output with RPAD function
Medium	N F S2	OPT.PLSQL.GEN_PLSQL.NFS2	NFS2: Do not format the output with LPAD function
Medium	N F S3	OPT.PLSQL.GEN_PLSQL.NFS3	NFS3: Do not format the output with RTRIM function
Medium	N F S4	OPT.PLSQL.GEN_PLSQL.NFS4	NFS4: Do not format the output with LTRIM function
Medium	Oracle Var	OPT.PLSQL.GEN_PLSQL.OracleVar	OracleVar: Do not use control variables outside EXCEPTION blocks
Medium	P P R	OPT.PLSQL.GEN_PLSQL.PPR	PPR: Use reference parameters
Medium	U H N	OPT.PLSQL.GEN_PLSQL.UHN	UHN: Use NOCOPY

Severity	Contrast rule	Engine rule ID	Description
Medium	Undef Case	OPT.PLSQL.GEN_PLSQL.UndefCase	UndefCase: Checks WHEN clauses are using the same control variable
Medium	Table Alias	OPT.PLSQL.MISC_PLSQL.TableAlias	TableAlias: Define an alias for each table
Medium	E X C	OPT.PLSQL.NOM_PLSQL.EXC	EXC: Naming convention for exceptions
Medium	E X C G	OPT.PLSQL.NOM_PLSQL.EXCG	EXCG: Naming convention for global exceptions in packages
Medium	F U N C	OPT.PLSQL.NOM_PLSQL.FUNC	FUNC: Correct format for functions
Medium	F U N C2	OPT.PLSQL.NOM_PLSQL.FUNC2	FUNC2: Naming convention for functions
Medium	F V C	OPT.PLSQL.NOM_PLSQL.FVC	FVC: Correct format for variables
Medium	I D T	OPT.PLSQL.NOM_PLSQL.IDT	IDT: Variable names prefix
Medium	P A Q	OPT.PLSQL.NOM_PLSQL.PAQ	PAQ: Correct format for packages
Medium	P R M	OPT.PLSQL.NOM_PLSQL.PRM	PRM: Specific naming convention for parameters
Medium	P R O C	OPT.PLSQL.NOM_PLSQL.PROC	PROC: Correct format for procedures
Medium	Avoid Percent	OPT.PLSQL.OYR_PLSQL.AvoidPercent	AvoidPercent: Warns about queries that use LIKE filters and '%' patterns
Medium	I C T	OPT.PLSQL.OYR_PLSQL.ICT	ICT: Put columns of the table to query
Medium	Too Much Or	OPT.PLSQL.OYR_PLSQL.TooMuchOr	TooMuchOr: Do not perform several OR checks over the same field
Medium	Suspicious Code	OPT.PLSQL.SEC.SuspiciousCode	SuspiciousCode: Potential malicious code.
Medium	Unqualified Item At Invoker Rights Routine	OPT.PLSQL.SEC.UnqualifiedItemAtInvokerRightsRoutine	UnqualifiedItemAtInvokerRightsRoutine: Unqualified database items in AUTHID CURRENT_USER routine.
Medium	User Controlled SQL Primary Key	OPT.PLSQL.SEC.UserControlledSQLPrimaryKey	UserControlledSQLPrimaryKey: Avoid using an user controlled Primary Key into a query
Medium	Hardcoded Credential	OPT.PLSQL.SEC.HardcodedCredential	HardcodedCredential: Use of Hard-coded Credentials

PowerScript Scan rules

Contrast Scan supports these rules for PowerScript.

Severity	Contrast rule	Engine rule ID	Description
Critical	Avoid Undoc Events	OPT.POWERSCRIPT.DOC_POWSCRT.AvoidUndocEvents	AvoidUndocEvents: Avoid undocumented events
Critical	Avoid Undoc Functions	OPT.POWERSCRIPT.DOC_POWSCRT.AvoidUndocFunctions	AvoidUndocFunctions: Avoid undocumented functions/subroutines
High	Comments Ratio In Events	OPT.POWERSCRIPT.DOC_POWSCRT.CommentsRatioInEvents	CommentsRatioInEvents: Avoid events with a very low comment/code ratio
High	Comments Ratio In Functions	OPT.POWERSCRIPT.DOC_POWSCRT.CommentsRatioInFunctions	CommentsRatioInFunctions: Avoid functions/subroutines with a very low comment/code ratio
High	Art Less5 Param	OPT.POWERSCRIPT.GEN_POWSCRT.ArtLess5Param	ArtLess5Param: Avoid artifacts with a lot of parameters
High	Art Without Group By	OPT.POWERSCRIPT.GEN_POWSCRT.ArtWithoutGroupBy	ArtWithoutGroupBy: Do not use 'Group By'

Severity	Contrast rule	Engine rule ID	Description
High	Art Without Subqueries	OPT.POWERSCRIPT.GEN_POWSCRT.ArtWithoutSubqueries	ArtWithoutSubqueries: Avoid subqueries
High	Dt Win Access DB	OPT.POWERSCRIPT.GEN_POWSCRT.DtWinAccessDB	DtWinAccessDB: Do not use queries directly over database
High	Dynamic SQL	OPT.POWERSCRIPT.GEN_POWSCRT.DynamicSQL	DynamicSQL: Avoid using dynamic SQL in scripts
High	Dynamic SQL4 Less Tables	OPT.POWERSCRIPT.GEN_POWSCRT.DynamicSQL4LessTables	DynamicSQL4LessTables: Avoid dynamic complex SQL queries
High	Overriding Event	OPT.POWERSCRIPT.GEN_POWSCRT.OverridingEvent	OverridingEvent: Avoid overriding Event
High	Queries4tables	OPT.POWERSCRIPT.GEN_POWSCRT.Queries4tables	Queries4tables: Avoid SELECT over too many tables
High	Queries9select Param	OPT.POWERSCRIPT.GEN_POWSCRT.Queries9selectParam	Queries9selectParam: Avoid complex SELECT clause
High	Win Too Many Mth	OPT.POWERSCRIPT.GEN_POWSCRT.WinTooManyMth	WinTooManyMth: Avoid 'Windows' with too many methods
High	Fan In	OPT.POWERSCRIPT.OYR_POWSCRT.FanIn	FanIn: Avoid calling many times the same artifact
High	Fan Out	OPT.POWERSCRIPT.OYR_POWSCRT.FanOut	FanOut: Avoid calling many artifacts from same one
High	Menu High Inheritance	OPT.POWERSCRIPT.OYR_POWSCRT.MenuHighInheritance	MenuHighInheritance: Avoid Window with too high level of inheritance
High	No Inheritance	OPT.POWERSCRIPT.OYR_POWSCRT.NoInheritance	NoInheritance: Avoid artifacts without inheritance relations
High	Win High Inheritance	OPT.POWERSCRIPT.OYR_POWSCRT.WinHighInheritance	WinHighInheritance: Avoid Window with too high level of inheritance
Info	Data Window Naming Convention	OPT.POWERSCRIPT.NOM_POWSCRT.DataWindowNamingConvention	DataWindowNamingConvention: DataWindow naming convention
Info	Global Func Naming Convention	OPT.POWERSCRIPT.NOM_POWSCRT.GlobalFuncNamingConvention	GlobalFuncNamingConvention: Global function naming convention
Info	Global Var Naming Convention	OPT.POWERSCRIPT.NOM_POWSCRT.GlobalVarNamingConvention	GlobalVarNamingConvention: Global variables naming convention
Info	Instance Var Naming Convention	OPT.POWERSCRIPT.NOM_POWSCRT.InstanceVarNamingConvention	InstanceVarNamingConvention: Instance variable naming convention
Info	Menu Naming Convention	OPT.POWERSCRIPT.NOM_POWSCRT.MenuNamingConvention	MenuNamingConvention: Menu naming convention
Info	Structure Naming Convention	OPT.POWERSCRIPT.NOM_POWSCRT.StructureNamingConvention	StructureNamingConvention: Structure naming convention
Info	User Event Naming Convention	OPT.POWERSCRIPT.NOM_POWSCRT.UserEventNamingConvention	UserEventNamingConvention: User event naming convention
Info	Window Naming Convention	OPT.POWERSCRIPT.NOM_POWSCRT.WindowNamingConvention	WindowNamingConvention: Window naming convention
Medium	Avoid Global Functions	OPT.POWERSCRIPT.GEN_POWSCRT.AvoidGlobalFunctions	AvoidGlobalFunctions: Avoid using global functions
Medium	Avoid Global Vars	OPT.POWERSCRIPT.GEN_POWSCRT.AvoidGlobalVars	AvoidGlobalVars: Avoid using global variables
Medium	Too Long Lines	OPT.POWERSCRIPT.GEN_POWSCRT.TooLongLines	TooLongLines: Avoid artifacts with lines too long

Python Scan rules

Contrast Scan supports these rules for Python.

Severity	Contrast rule	Engine rule ID	Description
Critical	Too Much Origins Allowed Rule	OPT.PYTHON.SECURITY.TooMuchOriginsAllowedRule	TooMuchOriginsAllowedRule: CORS (Cross-origin resource sharing) too br
Critical	Missing Browser Xss Filter	OPT.PYTHON.DJANGO.MissingBrowserXssFilter	MissingBrowserXssFilter: Secure brow filter
Critical	Code Injection	OPT.PYTHON.SECURITY.CodeInjection	CodeInjection: Avoid non-neutralized controlled input in dynamic code eval
Critical	Command Injection	OPT.PYTHON.SECURITY.CommandInjection	CommandInjection: Improper Neutrali Special Elements used in an OS Com ('OS Command Injection')
Critical	Connection String Parameter Pollution	OPT.PYTHON.SECURITY.ConnectionStringParameterPollution	ConnectionStringParameterPollution: Connection string polluted with untrus
Critical	Cross Site Scripting	OPT.PYTHON.SECURITY.CrossSiteScripting	CrossSiteScripting: Improper Neutraliz Input During Web Page Generation ('Scripting')
Critical	DoS Regexp	OPT.PYTHON.SECURITY.DoSRegexp	DoSRegexp: Potential denial-of-service through malicious regular expression
Critical	JSON Injection	OPT.PYTHON.SECURITY.JSONInjection	JSONInjection: Avoid using non-neutr user-controlled input into JSON entitie Injection
Critical	Ldap Injection	OPT.PYTHON.SECURITY.LdapInjection	LdapInjection: Avoid non-neutralized controlled input in LDAP search filters
Critical	Mail Command Injection	OPT.PYTHON.SECURITY.MailCommandInjection	MailCommandInjection: Mail Command Injection
Critical	Memcached Injection	OPT.PYTHON.SECURITY.MemcachedInjection	MemcachedInjection: Avoid non-neutr user-controlled input to be stored into
Critical	No SQL Injection	OPT.PYTHON.SECURITY.NoSQLInjection	NoSQLInjection: Improper neutralizati special elements in data query logic (l injection)
Critical	SQL Injection	OPT.PYTHON.SECURITY.SqlInjection	SqlInjection: Avoid SQL code formed non neutralized user input (vulnerable Injection attacks)
Critical	Stored Cross Site Scripting	OPT.PYTHON.SECURITY.StoredCrossSiteScripting	StoredCrossSiteScripting: Improper Neutralization of Input During Web Pa Generation ('Cross-site Scripting')
Critical	Xpath Injection	OPT.PYTHON.SECURITY.XpathInjection	XpathInjection: Avoid XPath expression with non neutralized user input
Critical	Xml Entity Injection	OPT.PYTHON.SECURITY.XmlEntityInjection	XmlEntityInjection: XML entity injection
Critical	Path Traversal	OPT.PYTHON.SECURITY.PathTraversal	PathTraversal: Avoid non-neutralized controlled input to be part of a pathna or directory) used in I/O operations
Critical	Password In Redirect Rule	OPT.PYTHON.SECURITY.PasswordInRedirectRule	PasswordInRedirectRule: Password Management - Password in Redirect
Critical	Hardcoded Crypto Key	OPT.PYTHON.SECURITY.HardcodedCryptoKey	HardcodedCryptoKey: Hardcoded cry keys
Critical	Non Random IV With CBC Mode	OPT.PYTHON.SECURITY.NonRandomIVWithCBCMode	NonRandomIVWithCBCMode: Not us Random IV with CBC Mode

Severity	Contrast rule	Engine rule ID	Description
Critical	Weak Cryptographic Hash In Settings	OPT.PYTHON.DJANGO.WeakCryptographicHashInSettings	WeakCryptographicHashInSettings: Weak cryptographic hashes cannot guarantee integrity
High	Insufficient Session Expiration Rule	OPT.PYTHON.SECURITY.InsufficientSessionExpirationRule	InsufficientSessionExpirationRule: Checks that session expiration interval is positive and does not exceed a limit
High	Cookie Based Sessions	OPT.PYTHON.DJANGO.CookieBasedSessions	CookieBasedSessions: Cookie-based sessions with a unsafe configuration
High	Insufficient Django Settings Session Expiration	OPT.PYTHON.DJANGO.InsufficientDjangoSettingsSessionExpiration	InsufficientDjangoSettingsSessionExpiration: Checks that session expiration interval is positive and does not exceed a limit
High	Cookie Poisoning	OPT.PYTHON.SECURITY.CookiePoisoning	CookiePoisoning: Cookie Poisoning
High	Cross Site Request Forgery	OPT.PYTHON.SECURITY.CrossSiteRequestForgery	CrossSiteRequestForgery: Cross-site request forgery (CSRF)
High	Dont Use Exec	OPT.PYTHON.SECURITY.DontUseExec	DontUseExec: Avoid using exec() function
High	Header Manipulation	OPT.PYTHON.SECURITY.HeaderManipulation	HeaderManipulation: Avoid including unvalidated data in HTTP response headers in Cookies
High	Http Parameter Pollution Rule	OPT.PYTHON.SECURITY.HttpParameterPollutionRule	HttpParameterPollutionRule: HTTP parameter pollution (HPP)
High	Log Forging	OPT.PYTHON.SECURITY.LogForging	LogForging: Unvalidated untrusted input in log
High	Open Redirect	OPT.PYTHON.SECURITY.OpenRedirect	OpenRedirect: Do not allow to control redirect used in a redirect by an unvalidated input
High	Resource Injection	OPT.PYTHON.SECURITY.ResourceInjection	ResourceInjection: Improper control of resource identifiers ("Resource Injection")
High	Server Side Request Forgery	OPT.PYTHON.SECURITY.ServerSideRequestForgery	ServerSideRequestForgery: Creation of requests from a vulnerable server using untrusted input (server side request forgery (SSRF))
High	Trust Boundary	OPT.PYTHON.SECURITY.TrustBoundary	TrustBoundary: Trust boundary violation
High	Unsafe Reflection	OPT.PYTHON.SECURITY.UnsafeReflection	UnsafeReflection: Use of Externally-Controlled Input to Select Classes or Code ('Unsafe Reflection')
High	Xml Injection	OPT.PYTHON.SECURITY.XmlInjection	XmlInjection: Avoid using non-neutralized controlled input when creating XML documents
High	Mass Assignment Attack	OPT.PYTHON.DJANGO.MassAssignmentAttack	MassAssignmentAttack: Insufficient form validation
High	Avoid Calling Magic Methods	OPT.PYTHON.MAINTAINABILITY.AvoidCallingMagicMethods	AvoidCallingMagicMethods: Avoid calling magic methods
High	Avoid Too Complex Functions	OPT.PYTHON.MAINTAINABILITY.AvoidTooComplexFunctions	AvoidTooComplexFunctions: Avoid too complex functions
High	Avoid Assignments To True Or False	OPT.PYTHON.RELIABILITY.AvoidAssignmentsToTrueOrFalse	AvoidAssignmentsToTrueOrFalse: Avoid assignments to True or False
High	Avoid Chained Comparisons Containing Equality	OPT.PYTHON.RELIABILITY.AvoidChainedComparisonsContainingEquality	AvoidChainedComparisonsContainingEquality: Avoid chained comparisons containing equality operator

Severity	Contrast rule	Engine rule ID	Description
High	Avoid Default Mutable Arguments	OPT.PYTHON.RELIABILITY.AvoidDefaultMutableArguments	AvoidDefaultMutableArguments: Avoid default mutable parameters
High	Init Cannot Be A Generator	OPT.PYTHON.RELIABILITY.InitCannotBeAGenerator	InitCannotBeAGenerator: __init__ method cannot be a generator
High	Invalid Open Mode	OPT.PYTHON.RELIABILITY.InvalidOpenMode	InvalidOpenMode: Invalid open() mode
High	Open Files Using With	OPT.PYTHON.RELIABILITY.OpenFilesUsingWith	OpenFilesUsingWith: Open files using with statement
High	Same Method And Field Names	OPT.PYTHON.RELIABILITY.SameMethodAndFieldNames	SameMethodAndFieldNames: Method and class fields should not to be different or same capitalization
High	Using Deprecated Module	OPT.PYTHON.RELIABILITY.UsingDeprecatedModule	UsingDeprecatedModule: Avoid using deprecated modules
High	Cookies In Security Decision	OPT.PYTHON.SECURITY.CookiesInSecurityDecision	CookiesInSecurityDecision: Reliance on Cookies without Validation and Integrity Checking in a Security Decision
High	Unhandled SSL Error Rule	OPT.PYTHON.SECURITY.UnhandledSSLERrorRule	UnhandledSSLERrorRule: Unhandled SSL exception
High	User Controlled SQL Primary Key	OPT.PYTHON.SECURITY.UserControlledSQLPrimaryKey	UserControlledSQLPrimaryKey: Avoid user controlled Primary Key into a query
High	Insecure Direct Object References	OPT.PYTHON.DJANGO.InsecureDirectObjectReferences	InsecureDirectObjectReferences: Check user authentication and/ or authorization before let him modifying a sensible system resource
High	Missing Function Level Access Control	OPT.PYTHON.DJANGO.MissingFunctionLevelAccessControl	MissingFunctionLevelAccessControl: Missing an authorization check when performing an action which requires authorization
High	Hardcoded Credential	OPT.PYTHON.SECURITY.HardcodedCredential	HardcodedCredential: Empty or hardcoded passwords may compromise system security in a way that cannot be easily remedied
High	Hardcoded Ip	OPT.PYTHON.SECURITY.HardcodedIp	HardcodedIp: Do not write IP addresses in code
High	Hardcoded Salt	OPT.PYTHON.SECURITY.HardcodedSalt	HardcodedSalt: Use of hardcoded salt
High	Insecure Transport	OPT.PYTHON.SECURITY.InsecureTransport	InsecureTransport: Insecure transport
High	Insufficient Key Size Rule	OPT.PYTHON.SECURITY.InsufficientKeySizeRule	InsufficientKeySizeRule: Weak cryptography insufficient key length
High	Server Insecure Transport	OPT.PYTHON.SECURITY.ServerInsecureTransport	ServerInsecureTransport: Insecure transport on HTTP servers
High	Weak Cryptographic Hash	OPT.PYTHON.SECURITY.WeakCryptographicHash	WeakCryptographicHash: Weak cryptographic hash
High	Weak Encryption Algorithm	OPT.PYTHON.SECURITY.WeakEncryptionAlgorithm	WeakEncryptionAlgorithm: Weak symmetric encryption algorithm
Info	Empty Docstring	OPT.PYTHON.MAINTAINABILITY.EmptyDocstring	EmptyDocstring: Empty docstring
Info	Import Top Of File	OPT.PYTHON.MAINTAINABILITY.ImportTopOfFile	ImportTopOfFile: Module level import of file
Info	Line Too Long	OPT.PYTHON.MAINTAINABILITY.LineTooLong	LineTooLong: Line too long
Info	Missing Docstring	OPT.PYTHON.MAINTAINABILITY.MissingDocstring	MissingDocstring: Missing docstring

Severity	Contrast rule	Engine rule ID	Description
Info	Multiple Imports One Line	OPT.PYTHON.MAINTAINABILITY.MultipleImportsOneLine	MultipleImportsOneLine: Multiple imports on one line
Info	Too Many Local Variables	OPT.PYTHON.MAINTAINABILITY.TooManyLocalVariables	TooManyLocalVariables: Too many local variables
Info	Unnecessary Semicolon	OPT.PYTHON.MAINTAINABILITY.UnnecessarySemicolon	UnnecessarySemicolon: Unnecessary semicolon
Low	Empty Sequences Are False	OPT.PYTHON.EFFICIENCY.EmptySequencesAreFalse	EmptySequencesAreFalse: Empty sequences are False
Low	Init Dictionaries With Literals	OPT.PYTHON.EFFICIENCY.InitDictionariesWithLiterals	InitDictionariesWithLiterals: Use literals to initialize dictionaries
Low	Potential Class Or Static Method	OPT.PYTHON.EFFICIENCY.PotentialClassOrStaticMethod	PotentialClassOrStaticMethod: Class methods not accessing instance fields must be static
Low	Avoid Assigning A Lambda	OPT.PYTHON.MAINTAINABILITY.AvoidAssigningALambda	AvoidAssigningALambda: Avoid assigning a lambda expression
Low	Avoid Commented Out Code	OPT.PYTHON.MAINTAINABILITY.AvoidCommentedOutCode	AvoidCommentedOutCode: Avoid commenting out code blocks
Low	Avoid Name Repetition In Comparisons	OPT.PYTHON.MAINTAINABILITY.AvoidNameRepetitionInComparisons	AvoidNameRepetitionInComparisons: Avoid unnecessary name repetition in equality comparisons
Low	Avoid Nested Empty Blocks	OPT.PYTHON.MAINTAINABILITY.AvoidNestedEmptyBlocks	AvoidNestedEmptyBlocks: Avoid unnecessary nested empty blocks
Low	Bad Identity Check	OPT.PYTHON.MAINTAINABILITY.BadIdentityCheck	BadIdentityCheck: Avoid using the [] {} identity check
Low	Bad Type Comparison	OPT.PYTHON.MAINTAINABILITY.BadTypeComparison	BadTypeComparison: Use isinstance() for a type comparison
Low	Blank Line At End Of File	OPT.PYTHON.MAINTAINABILITY.BlankLineAtEndOfFile	BlankLineAtEndOfFile: It should be one blank line at the end of the file
Low	Blank Line Contains Whitespace	OPT.PYTHON.MAINTAINABILITY.BlankLineContainsWhitespace	BlankLineContainsWhitespace: Blank lines should not contain whitespaces
Low	Blank Lines Surrounding Function Or Class	OPT.PYTHON.MAINTAINABILITY.BlankLinesSurroundingFunctionOrClass	BlankLinesSurroundingFunctionOrClass: Module level functions and class definitions should be surrounded with two lines
Low	Blank Line Surrounding Methods	OPT.PYTHON.MAINTAINABILITY.BlankLineSurroundingMethods	BlankLineSurroundingMethods: Class methods should be surrounded with a single blank line
Low	Loops Else Clause Always Executed	OPT.PYTHON.MAINTAINABILITY.LoopsElseClauseAlwaysExecuted	LoopsElseClauseAlwaysExecuted: Loop else clause is always executed
Low	Merge Simple Conditional Branches	OPT.PYTHON.MAINTAINABILITY.MergeSimpleConditionalBranches	MergeSimpleConditionalBranches: Simple conditional statements should be merged
Low	Remove Statements After Jump	OPT.PYTHON.MAINTAINABILITY.RemoveStatementsAfterJump	RemoveStatementsAfterJump: Statements after a jump are dead code
Low	Simplify Repetitive Unequal Checks	OPT.PYTHON.MAINTAINABILITY.SimplifyRepetitiveUnequalChecks	SimplifyRepetitiveUnequalChecks: Simplify repetitive unequal checks
Low	Trailing Whitespace	OPT.PYTHON.MAINTAINABILITY.TrailingWhitespace	TrailingWhitespace: Avoid trailing whitespaces

Severity	Contrast rule	Engine rule ID	Description
Low	Use Chained Comparisons	OPT.PYTHON.MAINTAINABILITY.UseChainedComparisons	UseChainedComparisons: Consider using chained comparisons
Low	Use Negative Index For Last Element	OPT.PYTHON.MAINTAINABILITY.UseNegativeIndexForLastElement	UseNegativeIndexForLastElement: Consider using negative indexes to access the positions of collections
Low	Use Proper Inequality Operator	OPT.PYTHON.PORTABILITY.UseProperInequalityOperator	UseProperInequalityOperator: Avoid using <=> inequality operator
Low	Avoid Pre Increment And Pre Decrement Operators	OPT.PYTHON.RELIABILITY.AvoidPreIncrementAndPreDecrementOperators	AvoidPreIncrementAndPreDecrementOperators: Avoid using the pre-increment and pre-decrement operators
Low	Avoid Using Return Outside Function	OPT.PYTHON.RELIABILITY.AvoidUsingReturnOutsideFunction	AvoidUsingReturnOutsideFunction: Avoid using the return statement outside a function
Low	Avoid Using Yield Outside Function	OPT.PYTHON.RELIABILITY.AvoidUsingYieldOutsideFunction	AvoidUsingYieldOutsideFunction: Avoid using the yield statement outside a function
Low	Handle F I X M E Tags	OPT.PYTHON.RELIABILITY.HandleFIXMETags	HandleFIXMETags: Handle FIXME tags
Low	No Exception Type Specified	OPT.PYTHON.RELIABILITY.NoExceptionTypeSpecified	NoExceptionTypeSpecified: No exception type specified
Low	Information Exposure Through Debug Log	OPT.PYTHON.SECURITY.InformationExposureThroughDebugLog	InformationExposureThroughDebugLog: Avoid exposing sensitive information through debug logs
Low	Password In Comments	OPT.PYTHON.SECURITY.PasswordInComments	PasswordInComments: Storing password details in plaintext anywhere in the system or system code can compromise security
Medium	Plaintext Storage In A Cookie Rule	OPT.PYTHON.SECURITY.PlaintextStorageInACookieRule	PlaintextStorageInACookieRule: Clear storage of Sensitive Information in a cookie
Medium	Unsafe Cookie	OPT.PYTHON.SECURITY.UnsafeCookie	UnsafeCookie: Generate server-side cookies with adequate security properties
Medium	Avoid Host Name Checks Rule	OPT.PYTHON.SECURITY.AvoidHostNameChecksRule	AvoidHostNameChecksRule: Avoid client-side hostname checks, that are not reliable due to DNS poisoning
Medium	Format String Injection Rule	OPT.PYTHON.SECURITY.FormatStringInjectionRule	FormatStringInjectionRule: Exclude user input from format strings
Medium	Serialization Injection	OPT.PYTHON.SECURITY.SerializationInjection	SerializationInjection: Deserialization of untrusted data
Medium	Avoid Unnecessary Materialization	OPT.PYTHON.EFFICIENCY.AvoidUnnecessaryMaterialization	AvoidUnnecessaryMaterialization: Use list iterator instead of materializing the list
Medium	Improve List Extension	OPT.PYTHON.EFFICIENCY.ImproveListExtension	ImproveListExtension: Do not use concatenation to extend lists
Medium	Not Using Items To Iterate Dictionary	OPT.PYTHON.EFFICIENCY.NotUsingItemsToIterateDictionary	NotUsingItemsToIterateDictionary: Do not use items() to iterate over a dictionary
Medium	Not Using Zip To Iterate Pair Of Lists	OPT.PYTHON.EFFICIENCY.NotUsingZipToIteratePairOfLists	NotUsingZipToIteratePairOfLists: Do not use zip() to iterate over a pair of lists
Medium	Avoid Too Deeply Nested Statements	OPT.PYTHON.MAINTAINABILITY.AvoidTooDeeplyNestedStatements	AvoidTooDeeplyNestedStatements: Avoid deeply nested statements

Severity	Contrast rule	Engine rule ID	Description
Medium	Cls As First Argument	OPT.PYTHON.MAINTAINABILITY.ClsAsFirstArgument	ClsAsFirstArgument: The first argument methods should be "cls"
Medium	Dead Code	OPT.PYTHON.MAINTAINABILITY.DeadCode	DeadCode: Avoid dead code
Medium	Maximum Module Lines	OPT.PYTHON.MAINTAINABILITY.MaximumModuleLines	MaximumModuleLines: Maximum module lines permitted
Medium	More Than A Statement Single Line	OPT.PYTHON.MAINTAINABILITY.MoreThanAStatementSingleLine	MoreThanAStatementSingleLine: Avoid more than one statement per line
Medium	Naming Conventions	OPT.PYTHON.MAINTAINABILITY.NamingConventions	NamingConventions: Follow PEP 8 naming conventions for Python elements
Medium	Self As First Argument	OPT.PYTHON.MAINTAINABILITY.SelfAsFirstArgument	SelfAsFirstArgument: The first argument instance methods should be "self"
Medium	Too Many Arguments	OPT.PYTHON.MAINTAINABILITY.TooManyArguments	TooManyArguments: Too much arguments in function, method or lambda
Medium	Too Many Statements	OPT.PYTHON.MAINTAINABILITY.TooManyStatements	TooManyStatements: Too high number of statements in a method
Medium	Unnecessary Pass Stmt	OPT.PYTHON.MAINTAINABILITY.UnnecessaryPassStmt	UnnecessaryPassStmt: Avoid using pass statement when is not necessary
Medium	Wildcard Import	OPT.PYTHON.MAINTAINABILITY.WildcardImport	WildcardImport: Avoid using * in import statements
Medium	Access Dictionary Element	OPT.PYTHON.PORTABILITY.AccessDictionaryElement	AccessDictionaryElement: Do not use [] to access dictionary elements
Medium	Avoid Deprecated Raising Exception Form	OPT.PYTHON.PORTABILITY.AvoidDeprecatedRaisingExceptionForm	AvoidDeprecatedRaisingExceptionForm: Avoid using the deprecated raising exception form
Medium	Avoid Print Statement	OPT.PYTHON.PORTABILITY.AvoidPrintStatement	AvoidPrintStatement: Avoid using print statement
Medium	Dont Use Backticks	OPT.PYTHON.PORTABILITY.DontUseBackticks	DontUseBackticks: Avoid using backticks
Medium	Hardcoded Absolute Path	OPT.PYTHON.PORTABILITY.HardcodedAbsolutePath	HardcodedAbsolutePath: Do not hardcode absolute paths
Medium	Property On An Old Style Class	OPT.PYTHON.PORTABILITY.PropertyOnAnOldStyleClass	PropertyOnAnOldStyleClass: Avoid using @property decorator on old-style class
Medium	Use New Class Style	OPT.PYTHON.PORTABILITY.UseNewClassStyle	UseNewClassStyle: Use the new class definition style
Medium	Avoid Assigning Functions Not Returning A Value	OPT.PYTHON.RELIABILITY.AvoidAssigningFunctionsNotReturningAValue	AvoidAssigningFunctionsNotReturningAValue: Avoid assigning functions not returning a value
Medium	Avoid Break And Continue Outside Loop	OPT.PYTHON.RELIABILITY.AvoidBreakAndContinueOutsideLoop	AvoidBreakAndContinueOutsideLoop: Avoid using the BREAK and CONTINUE statements outside a LOOP
Medium	Avoid Capturing Generic Exception	OPT.PYTHON.RELIABILITY.AvoidCapturingGenericException	AvoidCapturingGenericException: Avoid capturing generic exceptions
Medium	Avoid Characters And Numerals Confusion	OPT.PYTHON.RELIABILITY.AvoidCharactersAndNumeralsConfusion	AvoidCharactersAndNumeralsConfusion: Avoid confusion between characters and numerals when used as name identifiers
Medium	Avoid Empty Except	OPT.PYTHON.RELIABILITY.AvoidEmptyExcept	AvoidEmptyExcept: Avoid using empty except clauses
Medium	Avoid Explicit Returns Init	OPT.PYTHON.RELIABILITY.AvoidExplicitReturnsInit	AvoidExplicitReturnsInit: Avoid returning into __init__

Severity	Contrast rule	Engine rule ID	Description
Medium	Avoid Using Return And Yield Together	OPT.PYTHON.RELIABILITY.AvoidUsingReturnAndYieldTogether	AvoidUsingReturnAndYieldTogether: Avoid using return inside a generator
Medium	Check Exit Method Signature	OPT.PYTHON.RELIABILITY.CheckExitMethodSignature	CheckExitMethodSignature: Check method signature
Medium	Duplicate Argument Name	OPT.PYTHON.RELIABILITY.DuplicateArgumentName	DuplicateArgumentName: Avoid using name for more than a function argument
Medium	Duplicated Field Name With Class	OPT.PYTHON.RELIABILITY.DuplicatedFieldNameWithClass	DuplicatedFieldNameWithClass: Avoid same name for attribute than the class
Medium	Future Import Is Not The First	OPT.PYTHON.RELIABILITY.FutureImportsNotTheFirst	FutureImportsNotTheFirst: Avoid to import <code>__future__</code> module in the middle of a module
Medium	Redefinition Into List Comprehension	OPT.PYTHON.RELIABILITY.RedefinitionIntoListComprehension	RedefinitionIntoListComprehension: Avoid variable redefinitions into a list comprehension
Medium	Static Method First Argument	OPT.PYTHON.RELIABILITY.StaticMethodFirstArgument	StaticMethodFirstArgument: The first argument of static methods should not be neither <code>cls</code> nor <code>self</code>
Medium	Unreachable Code	OPT.PYTHON.RELIABILITY.UnreachableCode	UnreachableCode: Avoid unreachable code
Medium	Execution After Redirect	OPT.PYTHON.SECURITY.ExecutionAfterRedirect	ExecutionAfterRedirect: Execution After Redirect (EAR)
Medium	Potential Infinite Loop	OPT.PYTHON.SECURITY.PotentialInfiniteLoop	PotentialInfiniteLoop: Loop with Unreachable Exit Condition ('Infinite Loop')
Medium	Unchecked Input In Loop Condition	OPT.PYTHON.SECURITY.UncheckedInputInLoopCondition	UncheckedInputInLoopCondition: Unchecked input in loop condition
Medium	Hardcoded Auth Data	OPT.PYTHON.SECURITY.HardcodedAuthData	HardcodedAuthData: Use of Hard-coded Credentials
Medium	Information Exposure Through Error Message	OPT.PYTHON.SECURITY.InformationExposureThroughErrorMessage	InformationExposureThroughErrorMessage: Avoid sensitive information exposure through error messages
Medium	Password In Configuration File	OPT.PYTHON.SECURITY.PasswordInConfigurationFile	PasswordInConfigurationFile: Use of password into configuration file
Medium	Insecure Randomness	OPT.PYTHON.SECURITY.InsecureRandomness	InsecureRandomness: Standard pseudo-random number generators cannot withstand cryptographic attacks

RPG4 Scan rules

Contrast Scan supports these rules for RPG4.

Severity	Contrast rule	Engine rule ID	Description
Critical	Connection String Parameter Pollution	OPT.RPG4.SEC.ConnectionStringParameterPollution	ConnectionStringParameterPollution: Connection string polluted with untrusted input
Critical	Cross Site Scripting	OPT.RPG4.SEC.CrossSiteScripting	CrossSiteScripting: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')

Severity	Contrast rule	Engine rule ID	Description
Critical	OS Command Injection	OPT.RPG4.SEC.OSCommandInjection	OSCommandInjection: Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
Critical	Process Control	OPT.RPG4.SEC.ProcessControl	ProcessControl: Avoid calling subprogram where its name could be controlled by user input
Critical	Regex Injection	OPT.RPG4.SEC.RegexInjection	RegexInjection: Prevent denial of service attack through malicious regular expression ('Regex Injection')
Critical	Resource Injection	OPT.RPG4.SEC.ResourceInjection	ResourceInjection: Improper Control of Resource Identifiers ('Resource Injection')
Critical	SQL Injection	OPT.RPG4.SEC.SqlInjection	SqlInjection: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
Critical	Alloc Heap Misuse	OPT.RPG4.REL.AllocHeapMisuse	AllocHeapMisuse: Check that allocated memory is properly freed
Critical	Path Manipulation	OPT.RPG4.SEC.PathManipulation	PathManipulation: External Control of File Name or Path
Critical	Read Record Before Update Delete	OPT.RPG4.SEC.ReadRecordBeforeUpdateDelete	ReadRecordBeforeUpdateDelete: A record UPDATE or DELETE operation must be preceded by a record read operation (CHAIN or READxxx)
Critical	Special Authority Granted	OPT.RPG4.SEC.SpecialAuthorityGranted	SpecialAuthorityGranted: Least privilege failure due to special authority granted
Critical	Unexpected Key Select	OPT.RPG4.SEC.UnexpectedKeySelect	UnexpectedKeySelect: Authorization Bypass Through User-Controlled SQL Primary Key
Critical	Check Crypto Return Code	OPT.RPG4.SEC.CheckCryptoReturnCode	CheckCryptoReturnCode: Validate return code for cryptographic operations
Critical	Hardcoded Crypto Key	OPT.RPG4.SEC.HardcodedCryptoKey	HardcodedCryptoKey: Use of Hard-coded Cryptographic Key
High	Ldap Injection	OPT.RPG4.SEC.LdapInjection	LdapInjection: Avoid non-neutralized user-controlled input in LDAP search filters.
High	Log Forging	OPT.RPG4.SEC.LogForging	LogForging: Improper Output Neutralization for Logs
High	Avoid Binary Declarations	OPT.RPG4.AvoidBinaryDeclarations	AvoidBinaryDeclarations: Avoid to declare variables of binary type
High	Const Params When Not Modified	OPT.RPG4.ConstParamsWhenNotModified	ConstParamsWhenNotModified: To use 'Const' in parameters of subprocedures that are not modified
High	Initialize Variables	OPT.RPG4.InitializeVariables	InitializeVariables: Initialize each variable in its declaration
High	Call Parameter Mismatch	OPT.RPG4.REL.CallParameterMismatch	CallParameterMismatch: Parameter mismatch in CALL
High	Pointer Arithmetic	OPT.RPG4.SEC.PointerArithmetic	PointerArithmetic: Avoid pointer arithmetic in RPG
High	Hardcoded Ip	OPT.RPG4.SEC.HardcodedIp	HardcodedIp: Do not write IP address in source code
High	No Active Debug Rule	OPT.RPG4.SEC.NoActiveDebugRule	NoActiveDebugRule: Information Exposure Through Debug Information
High	Position Before Read File	OPT.RPG4.SEC.PositionBeforeReadFile	PositionBeforeReadFile: Every READE command must be preceded by SETLL
High	Sensitive Security Api Call	OPT.RPG4.SEC.SensitiveSecurityApiCall	SensitiveSecurityApiCall: Call to sensitive Security API element.

Severity	Contrast rule	Engine rule ID	Description
High	Insecure Randomness	OPT.RPG4.SEC.InsecureRandomness	InsecureRandomness: Standard pseudo-random number generators cannot withstand cryptographic attacks
High	Insufficient Key Size	OPT.RPG4.SEC.InsufficientKeySize	InsufficientKeySize: Weak cryptography, insufficient key length
High	Weak Crypto Hash	OPT.RPG4.SEC.WeakCryptoHash	WeakCryptoHash: Weak cryptographic hashes cannot guarantee data integrity
High	Weak Encryption Algorithm	OPT.RPG4.SEC.WeakEncryptionAlgorithm	WeakEncryptionAlgorithm: Weak encryption algorithm
Info	Avoid Capital Specification	OPT.RPG4.AvoidCapitalSpecification	AvoidCapitalSpecification: Use small letters for certain sections
Info	Avoid Hard Coding	OPT.RPG4.AvoidHardCoding	AvoidHardCoding: Declare constants instead of string and integer literals
Info	Avoid Old Opcodes	OPT.RPG4.AvoidOldOpcodes	AvoidOldOpcodes: Avoid old computations in RPG IV
Info	Capital Logical Operator	OPT.RPG4.CapitalLogicalOperator	CapitalLogicalOperator: Logical operators must be coded in upper-case
Info	Comments In Program	OPT.RPG4.CommentsInProgram	CommentsInProgram: Place comments in the header and on each procedure declaration of a RPG program
Info	Declarations Order	OPT.RPG4.DeclarationsOrder	DeclarationsOrder: Elements declaration order in RPG programs / procedures
Info	Format H Specifications	OPT.RPG4.FormatHSpecifications	FormatHSpecifications: Control specifications must begin at column 8
Info	Store Copy Prototype	OPT.RPG4.StoreCopyPrototype	StoreCopyPrototype: To copy prototypes when it is necessary
Info	Use B I F Instead Op Code	OPT.RPG4.UseBIFInsteadOpCode	UseBIFInsteadOpCode: Use built-in functions instead of operations codes
Info	Use Eval For String Manipulation	OPT.RPG4.UseEvalForStringManipulation	UseEvalForStringManipulation: It is preferable to use the EVAL statement of free-format
Info	Use Select Instead Cas Or Nested If	OPT.RPG4.UseSelectInsteadCasOrNestedIf	UseSelectInsteadCasOrNestedIf: Avoid complex IF ... ELSEIF, or CASxx
Info	Password In Comment	OPT.RPG4.SEC.PasswordInComment	PasswordInComment: Avoid placing passwords and other sensitive info in code comments
Low	Avoid Blocked Records	OPT.RPG4.AvoidBlockedRecords	AvoidBlockedRecords: Use (N) option to read records
Low	Avoid Dangerous Conditional Sentences	OPT.RPG4.AvoidDangerousConditionalSentences	AvoidDangerousConditionalSentences: Do not use GOTO / TAG, CABXX and COMP statements
Low	Avoid Display Operation	OPT.RPG4.AvoidDisplayOperation	AvoidDisplayOperation: Not to use the operation Display
Low	Avoid From Column In Data Fields	OPT.RPG4.AvoidFromColumnInDataFields	AvoidFromColumnInDataFields: Avoid using columns to indicate the beginning of a field
Low	Avoid Obsolete Loops	OPT.RPG4.AvoidObsoleteLoops	AvoidObsoleteLoops: Do not use IFxx, WHENxx, DOUxx and DOWxx operations
Low	Avoid Read E In Loops	OPT.RPG4.AvoidReadEInLoops	AvoidReadEInLoops: Do not use READE operations within loops
Low	Avoid Read P Read Pe	OPT.RPG4.AvoidReadPReadPe	AvoidReadPReadPe: Avoid READP and READPE
Low	Avoid Special Chars	OPT.RPG4.AvoidSpecialChars	AvoidSpecialChars: Not to use special characters to define variables

Severity	Contrast rule	Engine rule ID	Description
Low	Built In Functions With Params	OPT.RPG4.BuiltInFunctionsWithParams	BuiltInFunctionsWithParams: Use %EOF(), %FOUND() and %EQUAL() with parameters
Low	Check Indicators Near Set	OPT.RPG4.CheckIndicatorsNearSet	CheckIndicatorsNearSet: Avoid excessive separation between assignment of an indicator and its usage
Low	Close Opened Files	OPT.RPG4.CloseOpenedFiles	CloseOpenedFiles: Close all the opened files
Low	End Block Instructions	OPT.RPG4.EndBlockInstructions	EndBlockInstructions: Use 'EndX' instructions instead of the generic 'End'
Low	Eval Instead Of Set Move	OPT.RPG4.EvalInsteadOfSetMove	EvalInsteadOfSetMove: Use EVAL instead of SETON, SETOFF, MOVE and MOVEA with indicators
Low	Include Procedure In Large Programs	OPT.RPG4.IncludeProcedureInLargePrograms	IncludeProcedureInLargePrograms: Avoid too long programs
Low	Large Procedures	OPT.RPG4.LargeProcedures	LargeProcedures: Avoid too large procedures
Low	Naming Conventions	OPT.RPG4.NamingConventions	NamingConventions: Some of naming conventions must be respected
Low	Overlay Instead Positional Notation	OPT.RPG4.OverlayInsteadPositionalNotation	OverlayInsteadPositionalNotation: Avoid positional notation
Low	Record Format In File Operations	OPT.RPG4.RecordFormatInFileOperations	RecordFormatInFileOperations: Use record-format in file operations
Low	Use Free Format Syntax	OPT.RPG4.UseFreeFormatSyntax	UseFreeFormatSyntax: Use free-format syntax when available
Low	Use Named Constants To Call Programs	OPT.RPG4.UseNamedConstantsToCallPrograms	UseNamedConstantsToCallPrograms: Use named constants with CALL and CALLB operations
Low	Avoid Debug Control Sentences	OPT.RPG4.AvoidDebugControlSentences	AvoidDebugControlSentences: Do not use DEBUG in control-specification statements
Medium	Format String Injection	OPT.RPG4.SEC.FormatStringInjection	FormatStringInjection: Exclude unsanitized user input from format strings
Medium	Avoid Calling Modules	OPT.RPG4.AvoidCallingModules	AvoidCallingModules: Avoid call modules
Medium	Avoid Declare Vbles In Calc Spec	OPT.RPG4.AvoidDeclareVblesInCalcSpec	AvoidDeclareVblesInCalcSpec: Avoid variable declarations in calculation specification
Medium	Constant Instead Array Table	OPT.RPG4.ConstantInsteadArrayTable	ConstantInsteadArrayTable: If an array field is assigned just once, declare it as a constant
Medium	Declare Date Properly	OPT.RPG4.DeclareDateProperly	DeclareDateProperly: Declare variables with the suitable type
Medium	Improve Function Keys	OPT.RPG4.ImproveFunctionKeys	ImproveFunctionKeys: Avoid indicators *Inkx
Medium	Include Inz Sr	OPT.RPG4.IncludeInzSr	IncludeInzSr: Include *InzSr subroutine in the main section of the RPG program
Medium	Naming Indicators	OPT.RPG4.NamingIndicators	NamingIndicators: To name all the declared indicators
Medium	Use Built In Instead Of Indicator	OPT.RPG4.UseBuiltInInsteadOfIndicator	UseBuiltInInsteadOfIndicator: Use built-in functions instead of indicators

Severity	Contrast rule	Engine rule ID	Description
Medium	Use Only Call P	OPT.RPG4.UseOnlyCallP	UseOnlyCallP: Not use CALL nor CALLB
Medium	Hardcoded Credential	OPT.RPG4.SEC.HardcodedCredential	HardcodedCredential: Use of Hard-coded Credentials
Medium	Poor Error Handling	OPT.RPG4.SEC.PoorErrorHandling	PoorErrorHandling: Ignoring error conditions may allow an attacker to induce unexpected behavior unnoticed
Medium	Privacy Violation	OPT.RPG4.SEC.PrivacyViolation	PrivacyViolation: Exposure of Private Information

Scala Scan rules

Contrast Scan supports these rules for Scala.

Severity	Contrast rule	Engine rule ID	Description
Critical	Too Broad CORS Policy	OPT.SCALA.SECURITY.TooBroadCORSPolicy	TooBroadCORSPolicy: Too much allowed origins in HTML5 Access-Control-Allow-Origin header
Critical	Too Much Origins Allowed	OPT.SCALA.SECURITY.TooMuchOriginsAllowed	TooMuchOriginsAllowedRule: CORS policy (Cross-origin resource sharing) too broad
Critical	Code Injection	OPT.SCALA.SECURITY.CodeInjection	CodeInjection: Avoid non-neutralized user-controlled input in dynamic code evaluation
Critical	Command Injection	OPT.SCALA.SECURITY.CommandInjection	CommandInjection: Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
Critical	Connection String Parameter Pollution	OPT.SCALA.SECURITY.ConnectionStringParameterPollution	ConnectionStringParameterPollution: Connection string polluted with untrusted input
Critical	Cross Site Scripting	OPT.SCALA.SECURITY.CrossSiteScripting	CrossSiteScripting: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
Critical	Http Splitting	OPT.SCALA.SECURITY.HttpSplitting	HttpSplitting: Improper Neutralization of CRLF Sequences in HTTP Headers ('HTTP Response Splitting')
Critical	JSON Injection	OPT.SCALA.SECURITY.JSONInjection	JSONInjection: Avoid using non-neutralized user-controlled input into JSON entities - JSON Injection
Critical	Ldap Injection	OPT.SCALA.SECURITY.LdapInjection	LdapInjection: Avoid non-neutralized user-controlled input in LDAP search filters
Critical	Mail Command Injection	OPT.SCALA.SECURITY.MailCommandInjection	MailCommandInjection: Mail Command Injection
Critical	No SQL Injection	OPT.SCALA.SECURITY.NoSQLInjection	NoSQLInjection: Improper neutralization of special elements in data query logic (NoSQL injection)
Critical	Process Control	OPT.SCALA.SECURITY.ProcessControl	ProcessControl: Library loaded from untrusted source
Critical	Regex Injection	OPT.SCALA.SECURITY.RegexInjection	RegexInjection: Prevent denial of service attack through malicious regular expression ('Regex Injection')
Critical	Same Origin Method Execution	OPT.SCALA.SECURITY.SameOriginMethodExecution	SameOriginMethodExecution: Same Origin Method Execution (SOME)
Critical	SQL Injection	OPT.SCALA.SECURITY.SqlInjection	SqlInjection: Avoid SQL code formed with non neutralized user input (vulnerable to SQL Injection attacks)
Critical	Xml Entity Injection	OPT.SCALA.SECURITY.XmlEntityInjection	XmlEntityInjection: XML entity injection
Critical	Accessibility Subversion	OPT.SCALA.SECURITY.AccessibilitySubversion	AccessibilitySubversionRule: Java access restriction subverted (Reflection)

Severity	Contrast rule	Engine rule ID	Description
Critical	Anonymous Ldap Bind	OPT.SCALA.SECURITY.AnonymousLdapBind	AnonymousLdapBind: Access Control - Anonymous LDAP Bind
Critical	Password In Redirect	OPT.SCALA.SECURITY.PasswordInRedirect	PasswordInRedirect: Password Management - Password in Redirect
Critical	Path Traversal	OPT.SCALA.SECURITY.PathTraversal	PathTraversal: Avoid non-neutralized user-controlled input composed in a pathname to a resource
Critical	Hardcoded Crypto Key	OPT.SCALA.SECURITY.HardcodedCryptoKey	HardcodedCryptoKey: Hardcoded cryptographic keys
Critical	Non Random IV With CBC Mode	OPT.SCALA.SECURITY.NonRandomIVWithCBCMode	NonRandomIVWithCBCMode: Not using a Random IV with CBC Mode
Critical	Weak Cryptographic Hash	OPT.SCALA.SECURITY.WeakCryptographicHash	WeakCryptographicHash: Weak cryptographic hash
Critical	Weak Encryption	OPT.SCALA.SECURITY.WeakEncryption	WeakEncryption: Weak symmetric encryption algorithm
High	Akka Security Misconfiguration	OPT.SCALA.SECURITY.AkkaSecurityMisconfiguration	AkkaSecurityMisconfiguration: Security misconfiguration in Akka framework.
High	Play Security Misconfiguration	OPT.SCALA.SECURITY.PlaySecurityMisconfiguration	PlaySecurityMisconfiguration: Security misconfiguration in Play framework.
High	Cross Site Request Forgery	OPT.SCALA.SECURITY.CrossSiteRequestForgery	CrossSiteRequestForgery: Cross-site request forgery (CSRF)
High	External Control Of Configuration Setting	OPT.SCALA.SECURITY.ExternalControlOfConfigurationSetting	ExternalControlOfConfigurationSetting: External Control of System or Configuration Setting
High	Http Parameter Pollution	OPT.SCALA.SECURITY.HttpParameterPollution	HttpParameterPollution: HTTP parameter pollution (HPP)
High	Open Redirect	OPT.SCALA.SECURITY.OpenRedirect	OpenRedirect: Do not allow to control the URL used in a redirect by an unvalidated input
High	Resource Injection	OPT.SCALA.SECURITY.ResourceInjection	ResourceInjection: Improper control of resource identifiers ("Resource Injection")
High	Server Side Request Forgery	OPT.SCALA.SECURITY.ServerSideRequestForgery	ServerSideRequestForgery: Creation of requests from a vulnerable server using untrusted input (server side request forgery SSRF)
High	Trust Boundary Violation	OPT.SCALA.SECURITY.TrustBoundaryViolation	TrustBoundaryViolation: Trust boundary violation
High	Unsafe Reflection	OPT.SCALA.SECURITY.UnsafeReflection	UnsafeReflection: Use of Externally-Controlled Input to Select Classes or Code ('Unsafe Reflection')
High	XPath Injection	OPT.SCALA.SECURITY.XPathInjection	XPathInjection: Improper Neutralization of Data within XPath Expressions ('XPath Injection')
High	Xslt Injection	OPT.SCALA.SECURITY.XsltInjection	XsltInjection: XML Injection (aka Blind XPath Injection)
High	Hardcoded Ip	OPT.SCALA.SECURITY.HardcodedIp	HardcodedIp: Do not write IP address in source code
High	Information Exposure Through Error Message	OPT.SCALA.SECURITY.InformationExposureThroughErrorMessage	InformationExposureThroughErrorMessage: Avoid sensitive information exposure through error messages
High	Cookies In Security Decision	OPT.SCALA.SECURITY.CookiesInSecurityDecision	CookiesInSecurityDecision: Reliance on Cookies without Validation and Integrity Checking in a Security Decision
High	User Controlled SQL Primary Key	OPT.SCALA.SECURITY.UserControlledSQLPrimaryKey	UserControlledSQLPrimaryKey: Avoid using an user controlled Primary Key into a query
High	Hardcoded Salt	OPT.SCALA.SECURITY.HardcodedSalt	HardcodedSalt: A hardcoded salt can compromise system security

Severity	Contrast rule	Engine rule ID	Description
High	Inadequate Padding	OPT.SCALA.SECURITY.InadequatePadding	InadequatePadding: Inadequate padding
High	Insecure Transport	OPT.SCALA.SECURITY.InsecureTransport	InsecureTransport: Insecure transport
High	Insufficient Key Size	OPT.SCALA.SECURITY.InsufficientKeySize	InsufficientKeySize: Weak cryptography, insufficient key length
Info	Log Forging	OPT.SCALA.SECURITY.LogForging	LogForging: Improper Output Neutralization for Logs
Medium	Plaintext Storage In A Cookie Rule	OPT.SCALA.SECURITY.PlaintextStorageInACookieRule	PlaintextStorageInACookieRule: Cleartext Storage of Sensitive Information in a Cookie
Medium	Unsafe Cookie	OPT.SCALA.SECURITY.UnsafeCookie	UnsafeCookie: Generate server-side cookies with adequate security properties
Medium	Avoid Host Name Checks	OPT.SCALA.SECURITY.AvoidHostNameChecks	AvoidHostNameChecks: Avoid checks on client-side hostname, that are not reliable due to DNS poisoning
Medium	Format String Injection	OPT.SCALA.SECURITY.FormatStringInjection	FormatStringInjection: Exclude unsanitized user input from format strings
Medium	Serialization Injection	OPT.SCALA.SECURITY.SerializationInjection	SerializationInjection: Deserialization of untrusted data
Medium	Hardcoded Username Password	OPT.SCALA.SECURITY.HardcodedUsernamePassword	HardcodedUsernamePassword: Use of Hard-coded Credentials
Medium	JSON P Hijacking	OPT.SCALA.SECURITY.JSONPHijacking	JSONPHijacking: Sensitive information exposed through JSONP
Medium	Password In Configuration File	OPT.SCALA.SECURITY.PasswordInConfigurationFile	PasswordInConfigurationFile: Use of credentials into configuration file
Medium	Avoid Native Calls	OPT.SCALA.SECURITY.AvoidNativeCalls	AvoidNativeCalls: Avoid calls from Scala to native (JNI) code
Medium	Plaintext Storage Of Password	OPT.SCALA.SECURITY.PlaintextStorageOfPassword	PlaintextStorageOfPassword: Plaintext Storage of a Password
Medium	Privacy Violation	OPT.SCALA.SECURITY.PrivacyViolation	PrivacyViolation: Exposure of Private Information ('Privacy Violation')
Medium	Serializable Class Containing Sensitive Data	OPT.SCALA.SECURITY.SerializableClassContainingSensitiveData	SerializableClassContainingSensitiveData: Serializable Class Containing Sensitive Data
Medium	Detail Error Leak	OPT.SCALA.SECURITY.DetailErrorLeak	DetailErrorLeakRule: Do not send detail error information to client
Medium	Execution After Redirect	OPT.SCALA.SECURITY.ExecutionAfterRedirect	ExecutionAfterRedirect: Execution After Redirect (EAR)
Medium	Potential Infinite Loop	OPT.SCALA.SECURITY.PotentialInfiniteLoop	PotentialInfiniteLoop: Loop with Unreachable Exit Condition ('Infinite Loop')
Medium	Unchecked Input In Loop Condition	OPT.SCALA.SECURITY.UncheckedInputInLoopCondition	UncheckedInputInLoopCondition: Unchecked input in loop condition
Medium	Insecure Randomness	OPT.SCALA.SECURITY.InsecureRandomness	InsecureRandomness: Standard pseudo-random number generators cannot withstand cryptographic attacks

SQL Scan rules

Contrast Scan supports these rules for SQL.

Severity	Contrast rule	Engine rule ID	Description
Critical	Detect Unaware Cross Joins	OPT.SQL.SQL_PB.DetectUnawareCrossJoins	DetectUnawareCrossJoins: Do not make "unnoticed" cartesian products in queries
Critical	Fetch And Declare Same Fields	OPT.SQL.SQL_PB.FetchAndDeclareSameFields	FetchAndDeclareSameFields: The number of fields to retrieve specified in the DECLARE CURSOR statement must be the same as the number of fields specified in the FETCH statement
Critical	Avoid Correlated Sub Selects	OPT.SQL.SQL_PERFORMANCE.AvoidCorrelatedSubSelects	AvoidCorrelatedSubSelects: Avoid nested SELECTs that use columns defined in outer SELECTs
Critical	Cursor For Update Where Current	OPT.SQL.SQL_PERFORMANCE.CursorForUpdateWhereCurrent	CursorForUpdateWhereCurrent: If a CURSOR is declared FOR UPDATE, DELETE and UPDATE must be used with the WHERE CURRENT specification
Critical	Dont Select Known Fields	OPT.SQL.SQL_PERFORMANCE.DontSelectKnownFields	DontSelectKnownFields: SELECT queries never should get fields used in the WHERE specification with {}
High	Avoid Scroll Cursor	OPT.SQL.AvoidScrollCursor	AvoidScrollCursor: Do not use SCROLL CURSOR clause
High	Check Updt Dlt Rowset	OPT.SQL.CheckUpdtDltRowset	CheckUpdtDltRowset: Do not use the FOR ROW n OF ROWSET clause in UPDATE or DELETE statements with MULTIROW option.
High	Check Insert Not Atomic	OPT.SQL.CheckInsertNotAtomic	CheckInsertNotAtomic: Do not use the NOT ATOMIC CONTINUE ON SQLEXCEPTION in SELECT statements with MULTIROW option
High	Avoid Dynamic SQL	OPT.SQL.AvoidDynamicSql	AvoidDynamicSql: Do not use dynamic SQL
High	Avoid Lock Table	OPT.SQL.AvoidLockTable	AvoidLockTable: Do not use LOCK TABLE statement
High	Avoid Numeric References In By Clauses	OPT.SQL.SQL_MAINTAINABILITY.AvoidNumericReferencesInByClauses	AvoidNumericReferencesInByClauses: Do not refer to column names with number indexes in * BY clauses
High	Use The As Keyword	OPT.SQL.SQL_MAINTAINABILITY.UseTheAsKeyword	UseTheAsKeyword: Use AS keyword when establishing an alias to tables
High	Avoid Declared Unopened Cursors	OPT.SQL.SQL_PERFORMANCE.AvoidDeclaredUnopenedCursors	AvoidDeclaredUnopenedCursors: If a CURSOR is declared, it must be opened
High	Avoid For Update Wait Forever	OPT.SQL.SQL_PERFORMANCE.AvoidForUpdateWaitForever	AvoidForUpdateWaitForever: No use FOR UPDATE without specifying NOWAIT or WAIT n
High	Avoid Opened Unclosed Cursors	OPT.SQL.SQL_PERFORMANCE.AvoidOpenedUnclosedCursors	AvoidOpenedUnclosedCursors: If a CURSOR is opened, it must be closed
High	Avoid Opened Unused Cursors	OPT.SQL.SQL_PERFORMANCE.AvoidOpenedUnusedCursors	AvoidOpenedUnusedCursors: If a CURSOR is opened, it must be used
High	Avoid Union	OPT.SQL.SQL_PERFORMANCE.AvoidUnion	AvoidUnion: Avoid selects with UNION
High	No Current Clause	OPT.SQL.SQL_PERFORMANCE.NoCurrentClause	NoCurrentClause: SQL queries with CURRENT clause are heavy-weighted and must be used only when necessary

Severity	Contrast rule	Engine rule ID	Description
Info	Avoid Concat Operator	OPT.SQL.AvoidConcatOperator	AvoidConcatOperator: Do not use concatenation operator
Info	Avoid Current Server	OPT.SQL.AvoidCurrentServer	AvoidCurrentServer: Do not use CURRENT SERVER
Info	Avoid Numeric Cursor	OPT.SQL.AvoidNumericCursor	AvoidNumericCursor: Incorrect nomenclature for cursor name
Info	Avoid Current Time	OPT.SQL.AvoidCurrentTime	AvoidCurrentTime: Do not use CURRENT TIME
Info	Avoid Current Date	OPT.SQL.AvoidCurrentDate	AvoidCurrentDate: Do not use CURRENT DATE
Info	Avoid Non Declared Cursor	OPT.SQL.AvoidNonDeclaredCursor	AvoidNonDeclaredCursor: Use of cursor not previously declared
Info	Avoid Current Timestamp	OPT.SQL.AvoidCurrentTimestamp	AvoidCurrentTimestamp: Do not use CURRENT TIMESTAMP, CURRENT DATE or CURRENT TIME
Info	Check Nulls In Select Count	OPT.SQL.CheckNullsInSelectCount	CheckNullsInSelectCount: Do not use null indicator in queries with COUNT function
Info	Use Null Indicator	OPT.SQL.UseNullIndicator	UseNullIndicator: Use null indicator in queries with MAX, MIN, AVG and SUM functions
Info	Avoid Non Qualified Joins	OPT.SQL.SQL_PORTABILITY.AvoidNonQualifiedJoins	AvoidNonQualifiedJoins: Make the type of join explicit
Low	Do Not Use Negation In Where	OPT.SQL.DoNotUseNegationInWhere	DoNotUseNegationInWhere: Do not use negation operator
Low	Avoid Whenever	OPT.SQL.AvoidWhenever	AvoidWhenever: Do not use WHENEVER clause
Low	Avoid Host Operations	OPT.SQL.AvoidHostOperations	AvoidHostOperations: Do not perform arithmetic operations in the WHERE clause
Low	Avoid Having	OPT.SQL.AvoidHaving	AvoidHaving: Do not use HAVING clause
Low	Check Simple Condition	OPT.SQL.CheckSimpleCondition	CheckSimpleCondition: Simplify WHERE clause conditions
Low	Avoid Qualified Tables In Queries	OPT.SQL.SQL_MAINTAINABILITY.AvoidQualifiedTablesInQueries	AvoidQualifiedTablesInQueries: Table names should not be qualified in queries
Low	One SQL Statement Per Line	OPT.SQL.SQL_MAINTAINABILITY.OneSQLStatementPerLine	OneSQLStatementPerLine: Only write a SQL statement per line
Low	Qualified Tables In Queries	OPT.SQL.SQL_MAINTAINABILITY.QualifiedTablesInQueries	QualifiedTablesInQueries: Every table referenced in the query must be qualified
Low	Avoid Insert Without Fields Specification	OPT.SQL.SQL_PB.AvoidInsertWithoutFieldsSpecification	AvoidInsertWithoutFieldsSpecification: Every INSERT statement must include the field specification (i.e : INSERT INTO table(column1,column2) VALUES (value1,value2))
Low	Avoid Group By	OPT.SQL.SQL_PERFORMANCE.AvoidGroupBy	AvoidGroupBy: Avoid to use GROUP BY clause
Medium	Avoid Case In Select	OPT.SQL.AvoidCaseInSelect	AvoidCaseInSelect: Do not use CASE clause

Severity	Contrast rule	Engine rule ID	Description
Medium	Check Number Tables	OPT.SQL.CheckNumberTables	CheckNumberTables: Do not code queries with more than one table
Medium	Fully Qualified Names In Columns	OPT.SQL.SQL_MAINTAINABILITY.FullyQualifiedNamesInColumns	FullyQualifiedNamesInColumns: Use qualified names when referring to column names
Medium	Avoid Natural Joins	OPT.SQL.SQL_PB.AvoidNaturalJoins	AvoidNaturalJoins: NATURAL JOINS are buggy and unmaintenable
Medium	Avoid Nested Selects	OPT.SQL.SQL_PERFORMANCE.AvoidNestedSelects	AvoidNestedSelects: Avoid nested selects
Medium	Avoid Queries On Many Tables	OPT.SQL.SQL_PERFORMANCE.AvoidQueriesOnManyTables	AvoidQueriesOnManyTables: Avoid JOIN queries referencing too many tables
Medium	Avoid Select Asterisk	OPT.SQL.SQL_PERFORMANCE.AvoidSelectAsterisk	AvoidSelectAsterisk: Do not use SELECT *
Medium	Avoid Too Many Joins	OPT.SQL.SQL_PERFORMANCE.AvoidTooManyJoins	AvoidTooManyJoins: Avoid queries with too many JOINS
Medium	Detect Implicit Joins	OPT.SQL.SQL_PERFORMANCE.DetectImplicitJoins	DetectImplicitJoins: Never use implicit JOINS
Medium	Prefer On Over Using	OPT.SQL.SQL_PORTABILITY.PreferOnOverUsing	PreferOnOverUsing: Replace Using clause for its equivalent On counterpart

SQLScript Scan rules

Contrast Scan supports these rules for SQLScript.

Severity	Contrast rule	Engine rule ID	Description
Critical	SQL Injection	OPT.HANA.SEC.SqlInjection	SqlInjection: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
Critical	Avoid Trace In Production	OPT.HANA.EFFICIENCY.AvoidTraceInProduction	AvoidTraceInProduction: Avoid TRACE in production code.
Critical	Deeply Nested Subqueries	OPT.HANA.EFFICIENCY.DeeplyNestedSubqueries	DeeplyNestedSubqueries: Deeply nested subqueries.
Critical	Use Of Calculation Engine Operator	OPT.HANA.EFFICIENCY.UseOfCalculationEngineOperator	UseOfCalculationEngineOperator: Use of HANA Calculation Engine Plan Operators (CE Functions).
Critical	Excessive Privileges Granted	OPT.HANA.SEC.ExcessivePrivilegesGranted	ExcessivePrivilegesGranted: Excessive privileges granted.
High	Non Trivial Subquery	OPT.HANA.EFFICIENCY.NonTrivialSubquery	NonTrivialSubquery: Non-trivial subqueries.
High	Select In Scalar Function	OPT.HANA.EFFICIENCY.SelectInScalarFunction	SelectInScalarFunction: SELECT ... INTO in scalar function.
High	Improper Parameter Usage	OPT.HANA.RELIABILITY.ImproperParameterUsage	ImproperParameterUsage: Improper parameter usage.
High	Forbidden Call	OPT.HANA.SEC.ForbiddenCall	ForbiddenCall: Call to unsafe or dangerous procedure / function.

Severity	Contrast rule	Engine rule ID	Description
Low	Language Not Specified	OPT.HANA.MAINTAINABILITY.LanguageNotSpecified	LanguageNotSpecified: LANGUAGE not specified.
Low	Unused Condition	OPT.HANA.MAINTAINABILITY.UnusedCondition	UnusedCondition: Unused Error Condition.
Low	Non Custom Error Code	OPT.HANA.RELIABILITY.NonCustomErrorCode	NonCustomErrorCode: Use of non-custom SQL Error Code.
Medium	Avoid Using Cursors	OPT.HANA.EFFICIENCY.AvoidUsingCursors	AvoidUsingCursors: Avoid Using Cursors.
Medium	Modification Statement In Loop	OPT.HANA.EFFICIENCY.ModificationStatementInLoop	ModificationStatementInLoop: Data modification statement in a loop.
Medium	Reads SQL Data Not Specified	OPT.HANA.EFFICIENCY.ReadsSqlDataNotSpecified	ReadsSqlDataNotSpecified: Specify READS SQL DATA for side-effect free procedures.
Medium	Unused Variable	OPT.HANA.EFFICIENCY.UnusedVariable	UnusedVariable: Unused local variable.
Medium	Use Of Uninitialized Var	OPT.HANA.RELIABILITY.UseOfUninitializedVar	UseOfUninitializedVar: Use of uninitialized variable.

Swift Scan rules

Contrast Scan supports these rules for Swift.

Severity	Contrast rule	Engine rule ID	Description
Critical	Code Injection	OPT.SWIFT.SECURITY.CodeInjection	CodeInjection: Avoid non-neutralized user controlled input in dynamic code evaluation
Critical	Command Injection	OPT.SWIFT.SECURITY.CommandInjection	CommandInjection: Improper Neutralization of Special Elements used in an OS Command (Command Injection')
Critical	Connection String Parameter Pollution	OPT.SWIFT.SECURITY.ConnectionStringParameterPollution	ConnectionStringParameterPollution: Connection string polluted with untrusted input
Critical	Cross Site Scripting	OPT.SWIFT.SECURITY.CrossSiteScripting	CrossSiteScripting: Improper Neutralization of Input During Web Page Generation ('Cross Site Scripting')
Critical	Header Manipulation	OPT.SWIFT.SECURITY.HeaderManipulation	HeaderManipulation: Avoid including untrusted data in HTTP response header or in Cookie
Critical	JSON Injection	OPT.SWIFT.SECURITY.JSONInjection	JSONInjection: Avoid using non-neutralized user controlled input into JSON entities - JSON Injection
Critical	Mail Command Injection	OPT.SWIFT.SECURITY.MailCommandInjection	MailCommandInjection: Mail Command Injection
Critical	No SQL Injection	OPT.SWIFT.SECURITY.NoSQLInjection	NoSQLInjection: Improper neutralization of user input elements in data query logic (NoSQL injection)
Critical	Regex Injection	OPT.SWIFT.SECURITY.RegexInjection	RegexInjection: Prevent denial of service through malicious regular expression ('Regex Injection')
Critical	SQL Injection	OPT.SWIFT.SECURITY.SqlInjection	SqlInjection: Avoid SQL code formed with non-neutralized user input (vulnerable to SQL attacks)
Critical	Xpath Injection	OPT.SWIFT.SECURITY.XpathInjection	XpathInjection: Avoid XPath expressions with non neutralized user input

Severity	Contrast rule	Engine rule ID	Description
Critical	Missing Password Field Masking	OPT.SWIFT.SECURITY.MissingPasswordFieldMasking	MissingPasswordFieldMasking: Password field is not masked
Critical	Avoid Dangerous Try	OPT.SWIFT.RELIABILITY.AvoidDangerousTry	AvoidDangerousTry: Avoid using try operator along with ! operator
Critical	Path Traversal	OPT.SWIFT.SECURITY.PathTraversal	PathTraversal: Avoid non-neutralized user controlled input composed in a pathname resource
Critical	Weak Cryptographic Hash	OPT.SWIFT.SECURITY.WeakCryptographicHash	WeakCryptographicHash: Weak cryptographic hashes cannot guarantee data integrity
Critical	Weak Cryptographic Hash Salt	OPT.SWIFT.SECURITY.WeakCryptographicHashSalt	WeakCryptographicHashSalt: Weak cryptographic salts cannot guarantee data integrity
Critical	Weak Cryptographic Key	OPT.SWIFT.SECURITY.WeakCryptographicKey	WeakCryptographicKey: Weak keys used for cryptographic purposes
Critical	Weak Encryption	OPT.SWIFT.SECURITY.WeakEncryption	WeakEncryption: Weak symmetric encryption algorithm
Critical	Weak Key Derivation Iteration	OPT.SWIFT.SECURITY.WeakKeyDerivationIteration	WeakKeyDerivationIteration: Too weak iteration count on key derivation
Critical	Weak Key Derivation Password	OPT.SWIFT.SECURITY.WeakKeyDerivationPassword	WeakKeyDerivationPassword: Empty or weak password used in key derivation
Critical	Weak Symmetric Encryption Initialization Vector	OPT.SWIFT.SECURITY.WeakSymmetricEncryptionInitializationVector	WeakSymmetricEncryptionInitializationVector: Weak encryption initialization data vector
Critical	Weak Symmetric Encryption Mode Of Operation	OPT.SWIFT.SECURITY.WeakSymmetricEncryptionModeOfOperation	WeakSymmetricEncryptionModeOfOperation: Do not use weak modes of operation with symmetric encryption
High	Http Parameter Pollution Rule	OPT.SWIFT.SECURITY.HttpParameterPollutionRule	HttpParameterPollutionRule: HTTP parameter pollution (HPP)
High	Log Forging	OPT.SWIFT.SECURITY.LogForging	LogForging: Improper Output Neutralization in Logs
High	Open Redirect	OPT.SWIFT.SECURITY.OpenRedirect	OpenRedirect: Do not allow to control the location used in a redirect by an unvalidated input
High	Resource Injection	OPT.SWIFT.SECURITY.ResourceInjection	ResourceInjection: Improper control of resource identifiers ("Resource Injection")
High	Unsafe Reflection	OPT.SWIFT.SECURITY.UnsafeReflection	UnsafeReflection: Use of Externally-Controlled Input to Select Classes or Code ('Unsafe Reflection')
High	URL Scheme Hijacking	OPT.SWIFT.SECURITY.URLSchemeHijacking	URLSchemeHijacking: URL scheme hijacking through user input
High	XML Entity Injection	OPT.SWIFT.SECURITY.XMLEntityInjection	XMLEntityInjection: XML entity injection
High	XML Injection	OPT.SWIFT.SECURITY.XMLInjection	XMLInjection: XML Injection (aka Blind XML Injection)
High	Hardcoded Ip	OPT.SWIFT.SECURITY.HardcodedIp	HardcodedIp: Do not write IP address in source code
High	Avoid Maximum Location Accuracy When Possible	OPT.SWIFT.EFFICIENCY.AvoidMaximumLocationAccuracyWhenPossible	AvoidMaximumLocationAccuracyWhenPossible: Avoid using by default the best location accuracy

Severity	Contrast rule	Engine rule ID	Description
High	Cache Date Formatters	OPT.SWIFT.EFFICIENCY.CacheDateFormatters	CacheDateFormatters: Cache a single instance of NSDateFormatter from NSDateFormatter types instead of creating multiple instances
High	Do Not Instantiate Temporal Objects Loops	OPT.SWIFT.EFFICIENCY.DoNotInstantiateTemporalObjectsLoops	DoNotInstantiateTemporalObjectsLoops: Avoid allocating temporal objects in loop bodies
High	Minimize Bluetooth Interaction	OPT.SWIFT.EFFICIENCY.MinimizeBluetoothInteraction	MinimizeBluetoothInteraction: Avoid using CBCentralManagerScanOptionAllowDuplicates as a scan option
High	Class Cyclomatic Complexity	OPT.SWIFT.MAINTAINABILITY.ClassCyclomaticComplexity	ClassCyclomaticComplexity: Avoid using classes with high cyclomatic complexity values
High	Dead Stores	OPT.SWIFT.MAINTAINABILITY.DeadStores	DeadStores: Bound local variable value is not used
High	Method Cyclomatic Complexity	OPT.SWIFT.MAINTAINABILITY.MethodCyclomaticComplexity	MethodCyclomaticComplexity: Avoid using methods with high cyclomatic complexity
High	Unused Local Var	OPT.SWIFT.MAINTAINABILITY.UnusedLocalVar	UnusedLocalVar: Unused local variable
High	Hardcoded Absolute Path	OPT.SWIFT.PORTABILITY.HardcodedAbsolutePath	HardcodedAbsolutePath: Do not hardcode absolute paths
High	Avoid Empty Catch Blocks	OPT.SWIFT.RELIABILITY.AvoidEmptyCatchBlocks	AvoidEmptyCatchBlocks: Avoid use empty catch blocks
High	User Controlled SQL Primary Key	OPT.SWIFT.SECURITY.UserControlledSQLPrimaryKey	UserControlledSQLPrimaryKey: Avoid using user controlled Primary Key into a query
High	Insecure Transport	OPT.SWIFT.SECURITY.InsecureTransport	InsecureTransport: Insecure transport
Low	Password In Comment Rule	OPT.SWIFT.SECURITY.PasswordInCommentRule	PasswordInCommentRule: Storing password details in plaintext anywhere in system or system code can compromise security
Low	Avoid Locks	OPT.SWIFT.EFFICIENCY.AvoidLocks	AvoidLocks: Avoid using locks
Low	Avoid Commented Out Code	OPT.SWIFT.MAINTAINABILITY.AvoidCommentedOutCode	AvoidCommentedOutCode: Avoid commenting out code blocks
Low	Functions Should Not Return Constants	OPT.SWIFT.MAINTAINABILITY.FunctionsShouldNotReturnConstants	FunctionsShouldNotReturnConstants: Functions shouldn't return the same constant value
Low	Review Useless Empty Blocks	OPT.SWIFT.MAINTAINABILITY.ReviewUselessEmptyBlocks	ReviewUselessEmptyBlocks: Avoid using if and conditional statements with empty blocks
Low	Unused Parameter	OPT.SWIFT.MAINTAINABILITY.UnusedParameter	UnusedParameter: Unused function parameter
Low	Only One Return	OPT.SWIFT.RELIABILITY.OnlyOneReturn	OnlyOneReturn: Too many return statements in function or method
Low	Unconditional Jump Statements	OPT.SWIFT.RELIABILITY.UnconditionalJumpStatements	UnconditionalJumpStatements: Wrong use of unconditional jump statements
Medium	Plaintext Storage In A Cookie Rule	OPT.SWIFT.SECURITY.PlaintextStorageInACookieRule	PlaintextStorageInACookieRule: Clear text storage of Sensitive Information in a Cookie
Medium	Unsafe Cookie	OPT.SWIFT.SECURITY.UnsafeCookie	UnsafeCookie: Generate server-side cookies with adequate security properties
Medium	Serialization Injection	OPT.SWIFT.SECURITY.SerializationInjection	SerializationInjection: Deserialization of untrusted data

Severity	Contrast rule	Engine rule ID	Description
Medium	String Format Injection	OPT.SWIFT.SECURITY.StringFormatInjection	StringFormatInjection: Exclude unsanitiz input from format strings
Medium	Hardcoded Username Password	OPT.SWIFT.SECURITY.HardcodedUsernamePassword	HardcodedUsernamePassword: Use of h coded Credentials
Medium	HTTP Response Caching Leak	OPT.SWIFT.SECURITY.HTTPResponseCachingLeak	HTTPResponseCachingLeak: HTTP sen responses being cached
Medium	Insecure Temporary File	OPT.SWIFT.SECURITY.InsecureTemporaryFile	InsecureTemporaryFile: Creating and us insecure temporary files can leave applic system data vulnerable to attack.
Medium	Keyboard Caching Leak	OPT.SWIFT.SECURITY.KeyboardCachingLeak	KeyboardCachingLeak: Sensitive data le through keyboard cache
Medium	Password In Configuration File	OPT.SWIFT.SECURITY.PasswordInConfigurationFile	PasswordInConfigurationFile: Use of cre into configuration file
Medium	Pasteboard Caching Leak	OPT.SWIFT.SECURITY.PasteboardCachingLeak	PasteboardCachingLeak: Sensitive data through the pasteboard caching mechan
Medium	Privacy Violation	OPT.SWIFT.SECURITY.PrivacyViolation	PrivacyViolation: Exposure of Private Info ('Privacy Violation')
Medium	Sensitive Core Data	OPT.SWIFT.SECURITY.SensitiveCoreData	SensitiveCoreData: Sensitive data stored CoreData('Privacy Violation')
Medium	Sensitive Data Accessed From Itunes	OPT.SWIFT.SECURITY.SensitiveDataAccessedFromItunes	SensitiveDataAccessedFromItunes: Exp Private Information ('Privacy Violation')
Medium	Sensitive SQL	OPT.SWIFT.SECURITY.SensitiveSQL	SensitiveSQL: Sensitive data stored into database('Privacy Violation')
Medium	Sensitive No SQL	OPT.SWIFT.SECURITY.SensitiveNoSQL	SensitiveNoSQL: Sensitive data stored in NoSQL database('Privacy Violation')
Medium	Sensitive User Defaults	OPT.SWIFT.SECURITY.SensitiveUserDefaults	SensitiveUserDefaults: Sensitive data st UserDefaults('Privacy Violation')
Medium	Serializable Class Containing Sensitive Data	OPT.SWIFT.SECURITY.SerializableClassContainingSensitiveData	SerializableClassContainingSensitiveData: Serializable Class Containing Sensitive D
Medium	Screen Caching Leak	OPT.SWIFT.SECURITY.ScreenCachingLeak	ScreenCachingLeak: Sensitive data leak through the screen caching mechanism i is backgrounded
Medium	Third Party Keyboard Allowed	OPT.SWIFT.SECURITY.ThirdPartyKeyboardAllowed	ThirdPartyKeyboardAllowed: Avoid expo sensitive data to third party keyboards.
Medium	Avoid Comparing Count To Zero	OPT.SWIFT.EFFICIENCY.AvoidComparingCountToZero	AvoidComparingCountToZero: Use isEmpty check is a collection is empty
Medium	Stop Scanning On Device Found	OPT.SWIFT.EFFICIENCY.StopScanningOnDeviceFound	StopScanningOnDeviceFound: Stop sca device has been already found
Medium	Vars Should Be Constants	OPT.SWIFT.EFFICIENCY.VarsShouldBeConstants	VarsShouldBeConstants: Vars that never should be constants
Medium	Avoid Excessive Nested Statements	OPT.SWIFT.MAINTAINABILITY.AvoidExcessiveNestedStatements	AvoidExcessiveNestedStatements: Avoid deeply nested statements

Severity	Contrast rule	Engine rule ID	Description
Medium	Avoid Same Class Field Names	OPT.SWIFT.MAINTAINABILITY.AvoidSameClassFieldNames	AvoidSameClassFieldNames: Avoid using same name for attribute than the class name
Medium	Avoid Same Method Field Names	OPT.SWIFT.MAINTAINABILITY.AvoidSameMethodFieldNames	AvoidSameMethodFieldNames: Method names and class fields should not be different only by capitalization
Medium	Avoid Many Parameters Function	OPT.SWIFT.MAINTAINABILITY.AvoidManyParametersFunction	AvoidManyParametersFunction: Too many arguments in function or method
Medium	Density Of Comments	OPT.SWIFT.MAINTAINABILITY.DensityOfComments	DensityOfComments: Source code must be properly commented
Medium	Nested Switch Statement	OPT.SWIFT.MAINTAINABILITY.NestedSwitchStatement	NestedSwitchStatement: Avoid to nest switch statements
Medium	One Statement Per Line	OPT.SWIFT.MAINTAINABILITY.OneStatementPerLine	OneStatementPerLine: Use only one statement per line
Medium	Unused Private Function	OPT.SWIFT.MAINTAINABILITY.UnusedPrivateFunction	UnusedPrivateFunction: Avoid unused private methods and constructors
Medium	Avoid Forced Type Conversion	OPT.SWIFT.RELIABILITY.AvoidForcedTypeConversion	AvoidForcedTypeConversion: Avoid using forced type conversion
Medium	Buffer Overflow	OPT.SWIFT.RELIABILITY.BufferOverflow	BufferOverflow: Potential memory corruption
Medium	Local Vars With Global Name	OPT.SWIFT.RELIABILITY.LocalVarsWithGlobalName	LocalVarsWithGlobalName: Same name for local and global variable
Medium	Potential Encoding Buffer Overflow	OPT.SWIFT.RELIABILITY.PotentialEncodingBufferOverflow	PotentialEncodingBufferOverflow: Potential memory corruption
Medium	Unreachable Code	OPT.SWIFT.RELIABILITY.UnreachableCode	UnreachableCode: Avoid unreachable code
Medium	Use Weak References With Delegate Protocols	OPT.SWIFT.RELIABILITY.UseWeakReferencesWithDelegateProtocols	UseWeakReferencesWithDelegateProtocols: Delegate protocols must be class-only
Medium	Avoid S M S	OPT.SWIFT.SECURITY.AvoidSMS	AvoidSMS: Avoid performing SMS-related operations
Medium	Biometric Without Message	OPT.SWIFT.SECURITY.BiometricWithoutMessage	BiometricWithoutMessage: User is asked for fingerprints without reason
Medium	Execution After Redirect	OPT.SWIFT.SECURITY.ExecutionAfterRedirect	ExecutionAfterRedirect: Execution After Redirect (EAR)
Medium	Missing Content Validation	OPT.SWIFT.SECURITY.MissingContentValidation	MissingContentValidation: Missing Content Validation
Medium	Potential Infinite Loop	OPT.SWIFT.SECURITY.PotentialInfiniteLoop	PotentialInfiniteLoop: Loop with Unreachable Condition ('Infinite Loop')
Medium	Server Trust Credential Check	OPT.SWIFT.SECURITY.ServerTrustCredentialCheck	ServerTrustCredentialCheck: Evaluate server certificate trust chain
Medium	Unchecked Input In Loop Condition	OPT.SWIFT.SECURITY.UncheckedInputInLoopCondition	UncheckedInputInLoopCondition: Unchecked input in loop condition

Transact-SQL Scan rules

Contrast Scan supports these rules for Transact-SQL.

Severity	Contrast rule	Engine rule ID	Description
Critical	Command Injection	OPT.TRANSACTSQL.SEC.CommandInjection	CommandInjection: Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
Critical	Sleep Injection	OPT.TRANSACTSQL.SEC.SleepInjection	SleepInjection: Denial of Service by externally controlled sleep time
Critical	SQL Injection	OPT.TRANSACTSQL.SEC.SqlInjection	SqlInjection: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
Critical	Avoid No Lock	OPT.TRANSACTSQL.AvoidNoLock	AvoidNoLock: Avoid NOLOCK table hint
Critical	Use Proper Transaction Isolation Level	OPT.TRANSACTSQL.UseProperTransactionIsolationLevel	UseProperTransactionIsolationLevel: Use proper Transaction Isolation Level
Critical	Too Broad Grant	OPT.TRANSACTSQL.SEC.TooBroadGrant	TooBroadGrant: Too broad privileges granted.
Critical	Weak Cryptographic Hash	OPT.TRANSACTSQL.SEC.WeakCryptographicHash	WeakCryptographicHash: Weak cryptographic hashes cannot guarantee data integrity
Critical	Weak Symmetric Encryption Algorithm	OPT.TRANSACTSQL.SEC.WeakSymmetricEncryptionAlgorithm	WeakSymmetricEncryptionAlgorithm: Weak symmetric encryption algorithm.
High	Avoid Email Hardcoded	OPT.TRANSACTSQL.AvoidEmailHardcoded	AvoidEmailHardcoded: Avoid hardcoded or in-comment emails in source code
High	Avoid IP Hardcoded	OPT.TRANSACTSQL.AvoidIPHardcoded	AvoidIPHardcoded: Do not write IP address in source code
High	Avoid Cross Joins	OPT.TRANSACTSQL.AvoidCrossJoins	AvoidCrossJoins: Avoid explicit or unwanted CROSS JOINS (cartesian products)
High	Avoid Delete Update Without Search Condition	OPT.TRANSACTSQL.AvoidDeleteUpdateWithoutSearchCondition	AvoidDeleteUpdateWithoutSearchCondition: Do not Use UPDATE / DELETE without search condition
High	Close Deallocate Cursors	OPT.TRANSACTSQL.CloseDeallocateCursors	CloseDeallocateCursors: Close/deallocate cursors and deallocate cursor variables in the same T-SQL scope where they are declared
High	Dead Variable Or Parameter	OPT.TRANSACTSQL.DeadVariableOrParameter	DeadVariableOrParameter: Looks for unused local variables and procedure/ function parameter.s
High	Use Exists Instead Of In	OPT.TRANSACTSQL.UseExistsInsteadOfIn	UseExistsInsteadOfIn: Avoid IN(subquery) if not necessary
High	Use Order By With Top	OPT.TRANSACTSQL.UseOrderByWithTop	UseOrderByWithTop: Use ORDER BY or GROUP BY clause in queries with TOP clause
High	Encrypt Information	OPT.TRANSACTSQL.EncryptInformation	EncryptInformation: Avoid including sensitive information in column names
High	Insecure Randomness	OPT.TRANSACTSQL.SEC.InsecureRandomness	InsecureRandomness: Standard pseudo-random number generators cannot withstand cryptographic attacks.
Info	Avoid Comparing With Null Constant	OPT.TRANSACTSQL.AvoidComparingWithNullConstant	AvoidComparingWithNullConstant: Use IS/IS NOT NULL instead of comparing against NULL constant
Info	Avoid Distinct	OPT.TRANSACTSQL.AvoidDistinct	AvoidDistinct: Avoid DISTINCT
Info	Avoid Large Composite Primary Keys	OPT.TRANSACTSQL.AvoidLargeCompositePrimaryKeys	AvoidLargeCompositePrimaryKeys: Limit columns in composite PRIMARY KEY
Info	Avoid Large Text Binary Objects	OPT.TRANSACTSQL.AvoidLargeTextBinaryObjects	AvoidLargeTextBinaryObjects: Do not store binary or image files (Binary Large Objects or BLOBs) inside the database

Severity	Contrast rule	Engine rule ID	Description
Info	Avoid Non Ansi Outer Join Syntax	OPT.TRANSACTSQL.AvoidNonAnsiOuterJoinSyntax	AvoidNonAnsiOuterJoinSyntax: Avoid non-ANSI outer join syntax * [] {}
Info	Avoid Nullable Char	OPT.TRANSACTSQL.AvoidNullableChar	AvoidNullableChar: Use the CHAR/NCHAR data type for a column only when the column is non-nullable
Info	Avoid Text Datatypes	OPT.TRANSACTSQL.AvoidTextDatatypes	AvoidTextDatatypes: Avoid TEXT, NTEXT or IMAGE datatypes for storing large textual/binary data
Info	Comment Tsql Code	OPT.TRANSACTSQL.CommentTsqlCode	CommentTsqlCode: Comment specific T-SQL elements
Info	Prefix Column Names With Table Name	OPT.TRANSACTSQL.PrefixColumnNamesWithTableName	PrefixColumnNamesWithTableName: Prefix column names with table name/alias in SQL statements referencing multiple tables
Info	Single Exit Point In Procedures	OPT.TRANSACTSQL.SingleExitPointInProcedures	SingleExitPointInProcedures: Code stored procedures, functions and triggers with a single exit point (RETURN)
Info	Use Ansi Join	OPT.TRANSACTSQL.UseAnsiJoin	UseAnsiJoin: Use ANSI-Standard Join clauses instead of the old style joins
Info	Use Set No Count On In Procedures	OPT.TRANSACTSQL.UseSetNoCountOnInProcedures	UseSetNoCountOnInProcedures: Use SET NOCOUNT ON at the beginning of SQL batches, stored procedures and triggers
Info	Use Standard Names	OPT.TRANSACTSQL.UseStandardNames	UseStandardNames: Follow naming conventions for names of database objects and T-SQL entities
Low	Avoid Cursors	OPT.TRANSACTSQL.AvoidCursors	AvoidCursors: Try to avoid server side cursors as much as possible
Low	Avoid Expressions That Prevent Index	OPT.TRANSACTSQL.AvoidExpressionsThatPreventIndex	AvoidExpressionsThatPreventIndex: Avoid expressions with columns in a query predicate that disable index search
Low	Avoid Goto	OPT.TRANSACTSQL.AvoidGoto	AvoidGoto: Do not use GOTO
Low	Avoid Group By Without Aggregation Functions	OPT.TRANSACTSQL.AvoidGroupByWithoutAggregationFunctions	AvoidGroupByWithoutAggregationFunctions: Do not use the GROUP BY clause without an aggregate function
Low	Avoid Recompile	OPT.TRANSACTSQL.AvoidRecompile	AvoidRecompile: Avoid operations that force a recompilation of an SQL batch or stored procedure / trigger
Low	Avoid Reserved Words In Identifiers	OPT.TRANSACTSQL.AvoidReservedWordsInIdentifiers	AvoidReservedWordsInIdentifiers: Do not use reserved words in identifiers
Low	Avoid Select Count From Table	OPT.TRANSACTSQL.AvoidSelectCountFromTable	AvoidSelectCountFromTable: Avoid SELECT COUNT to fetch number of rows in table
Low	Avoid System Prefixes	OPT.TRANSACTSQL.AvoidSystemPrefixes	AvoidSystemPrefixes: Do not use sp_ or fn_ as prefix for stored procedures and functions
Low	Explicit Column Names In Insert	OPT.TRANSACTSQL.ExplicitColumnNamesInInsert	ExplicitColumnNamesInInsert: Identify Column Names in INSERT statements
Low	Length With Varchar Types	OPT.TRANSACTSQL.LengthWithVarcharTypes	LengthWithVarcharTypes: Specify explicit length with character and binary data types
Low	Precision Scale With Decimal Numeric	OPT.TRANSACTSQL.PrecisionScaleWithDecimalNumeric	PrecisionScaleWithDecimalNumeric: Specify explicit precision and scale with DECIMAL and NUMERIC data types
Medium	Avoid Dynamic SQL	OPT.TRANSACTSQL.AvoidDynamicSql	AvoidDynamicSql: Avoid dynamic SQL statements as much as possible

Severity	Contrast rule	Engine rule ID	Description
Medium	Avoid Deprecated Features	OPT.TRANSACTSQL.AvoidDeprecatedFeatures	AvoidDeprecatedFeatures: Avoid deprecated features
Medium	Avoid Exact Or Overlapping Indexes	OPT.TRANSACTSQL.AvoidExactOrOverlappingIndexes	AvoidExactOrOverlappingIndexes: Avoid indexes with same or overlapping columns
Medium	Avoid Like Patterns Table Scan	OPT.TRANSACTSQL.AvoidLikePatternsTableScan	AvoidLikePatternsTableScan: Avoid patterns in LIKE that disable index usage and forces a table/index scan
Medium	Avoid Negative Operator	OPT.TRANSACTSQL.AvoidNegativeOperator	AvoidNegativeOperator: Avoid using <> or ! {}
Medium	Avoid Select Asterisk	OPT.TRANSACTSQL.AvoidSelectAsterisk	AvoidSelectAsterisk: Avoid * in SELECT
Medium	Avoid Too Many Joins	OPT.TRANSACTSQL.AvoidTooManyJoins	AvoidTooManyJoins: Avoid queries with too many joined tables
Medium	Avoid Trigger Return Data	OPT.TRANSACTSQL.AvoidTriggerReturnData	AvoidTriggerReturnData: Avoid returning results in triggers
Medium	Check Error After Data Manipulation	OPT.TRANSACTSQL.CheckErrorAfterDataManipulation	CheckErrorAfterDataManipulation: Check result from INSERT / UPDATE / DELETE using @@ERROR or @@ROWCOUNT or in TRY ... CATCH
Medium	Prefer Union All Over Union	OPT.TRANSACTSQL.PreferUnionAllOverUnion	PreferUnionAllOverUnion: Prefer UNION ALL over UNION
Medium	Forbidden Call	OPT.TRANSACTSQL.SEC.ForbiddenCall	ForbiddenCall: Dangerous procedure / function called.
Medium	User Controlled SQL Primary Key	OPT.TRANSACTSQL.SEC.UserControlledSQLPrimaryKey	UserControlledSQLPrimaryKey: Avoid using an user controlled Primary Key into a query

VB.NET Scan rules

Contrast Scan supports these rules for VB.NET.

Severity	Contrast rule	Engine rule ID	Description
Critical	Too Much Origins Allowed	OPT.VBNET.TooMuchOriginsAllowed	TooMuchOriginsAllowed: CORS policy (Cross origin resource sharing) too broad
Critical	Code Injection	OPT.VBNET.CodeInjection	CodeInjection: Improper Control of Generated Code ('Code Injection')
Critical	Code Injection With Deserialization	OPT.VBNET.CodeInjectionWithDeserialization	CodeInjectionWithDeserialization: Dynamic injection during object deserialization
Critical	Command Injection	OPT.VBNET.CommandInjection	CommandInjection: Improper Neutralization of Special Elements used in an OS Command ('Command Injection')
Critical	Cross Site Scripting	OPT.VBNET.CrossSiteScripting	CrossSiteScripting: Improper Neutralization of Input During Web Page Generation ('Cross-Site Scripting')
Critical	DoS Regexp	OPT.VBNET.DoSRegexp	DoSRegexp: Prevent denial of service attack through malicious regular expression
Critical	Ldap Injection	OPT.VBNET.LdapInjection	LdapInjection: Avoid non-neutralized user-controlled input in LDAP search filters
Critical	Connection String Parameter Pollution	OPT.VBNET.SEC.ConnectionStringParameterPollution	ConnectionStringParameterPollution: Connection string polluted with untrusted input

Severity	Contrast rule	Engine rule ID	Description
Critical	Http Parameter Pollution	OPT.VBNET.SEC.HttpParameterPollution	HttpParameterPollution: HTTP parameter pollution (HPP)
Critical	Http Splitting Rule	OPT.VBNET.SEC.HttpSplittingRule	HttpSplittingRule: Improper neutralization of CR/LF Sequences in HTTP headers
Critical	Mail Command Injection	OPT.VBNET.SEC.MailCommandInjection	MailCommandInjection: Mail Command Injection
Critical	No SQL Injection	OPT.VBNET.SEC.NoSQLInjection	NoSQLInjection: Improper neutralization of elements in data query logic (NoSQL injection)
Critical	Process Control	OPT.VBNET.SEC.ProcessControl	ProcessControl: Do not load executables or libraries from untrusted sources
Critical	Registry Manipulation	OPT.VBNET.SEC.RegistryManipulation	RegistryManipulation: Registry manipulation
Critical	Server Side Request Forgery	OPT.VBNET.ServerSideRequestForgery	ServerSideRequestForgery: Server-Side Request Forgery (SSRF)
Critical	SQL Injection	OPT.VBNET.SqlInjection	SqlInjection: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
Critical	Stored Cross Site Scripting	OPT.VBNET.StoredCrossSiteScripting	StoredCrossSiteScripting: Web content generated from improper sanitized database data and escaped output (Stored Cross-site Scripting)
Critical	MVC Non Action Public Methods	OPT.VBNET.MVCNonActionPublicMethods	MVCNonActionPublicMethods: Protect public methods that are not action methods in controllers
Critical	Path Traversal	OPT.VBNET.PathTraversal	PathTraversal: Avoid non-neutralized user-controlled input composed in a pathname to access a resource
Critical	Accessibility Subversion Rule	OPT.VBNET.SEC.AccessibilitySubversionRule	AccessibilitySubversionRule: .Net access restriction subverted (Reflection)
Critical	Anonymous Ldap Bind	OPT.VBNET.SEC.AnonymousLdapBind	AnonymousLdapBind: Access Control - Anonymous LDAP Bind
Critical	Dangerous File Upload	OPT.VBNET.SEC.DangerousFileUpload	DangerousFileUpload: Unrestricted Upload with Dangerous Type
Critical	Static Database Connection	OPT.VBNET.SEC.StaticDatabaseConnection	StaticDatabaseConnection: Static database connection / session
Critical	Temporary Files Left	OPT.VBNET.SEC.TemporaryFilesLeft	TemporaryFilesLeft: Temporary files not deleted
Critical	COM With Param Constructors	OPT.VBNET.VBnet.COMWithParamConstructors	COMWithParamConstructors: Avoid COM vtbl attribute in parameterized constructors
Critical	Dispose Finalize Throws Ex	OPT.VBNET.VBnet.DisposeFinalizeThrowsEx	DisposeFinalizeThrowsEx: Avoid throwing exception in constructors, Dispose or Finalize
Critical	Dispose Objects Before Losing Scope	OPT.VBNET.VBnet.DisposeObjectsBeforeLosingScope	DisposeObjectsBeforeLosingScope: Dispose objects before losing scope
Critical	Do Not Dispose Objects Multiple Times	OPT.VBNET.VBnet.DoNotDisposeObjectsMultipleTimes	DoNotDisposeObjectsMultipleTimes: Possible multiple calls to 'Dispose' over an object
Critical	Do Not Use Idle Process Priority	OPT.VBNET.VBnet.DoNotUseIdleProcessPriority	DoNotUseIdleProcessPriority: Do not use idle process priority
Critical	Mark I Serializable Types With Serializable	OPT.VBNET.VBnet.MarkISerializableTypesWithSerializable	MarkISerializableTypesWithSerializable: Specify the Serializable attribute in ISerializable classes
Critical	Mark Windows Forms Entry Points With Sta Thread	OPT.VBNET.VBnet.MarkWindowsFormsEntryPointsWithStaThread	MarkWindowsFormsEntryPointsWithStaThread: Mark Windows Forms entry points with STA Thread

Severity	Contrast rule	Engine rule ID	Description
Critical	Weak Cryptographic Hash	OPT.VBNET.WeakCryptographicHash	WeakCryptographicHash: Weak cryptographic hash
Critical	Weak Key Size	OPT.VBNET.WeakKeySize	WeakKeySize: Weak cryptography, insufficient key length
Critical	Weak Symmetric Encryption Algorithm	OPT.VBNET.WeakSymmetricEncryptionAlgorithm	WeakSymmetricEncryptionAlgorithm: Weak symmetric encryption algorithm
Critical	Weak Symmetric Encryption Mode Of Operation	OPT.VBNET.WeakSymmetricEncryptionModeOfOperation	WeakSymmetricEncryptionModeOfOperation: not use weak modes of operation with symmetric encryption
High	Cross Site Request Forgery	OPT.VBNET.CrossSiteRequestForgery	CrossSiteRequestForgery: Cross-Site Request Forgery (CSRF)
High	JSON Injection	OPT.VBNET.JSONInjection	JSONInjection: Avoid using non-neutralized controlled input in JSON entities
High	MVC Prevent Overposting Model Definition	OPT.VBNET.MVCPreventOverpostingModelDefinition	MVCPreventOverpostingModelDefinition: Prevent over-posting attacks in model definition
High	MVC Prevent Underposting Model Composition	OPT.VBNET.MVCPreventUnderpostingModelComposition	MVCPreventUnderpostingModelComposition: Prevent under-posting attacks in model composition
High	MVC Prevent Underposting Model Definition	OPT.VBNET.MVCPreventUnderpostingModelDefinition	MVCPreventUnderpostingModelDefinition: Prevent under-posting attacks in model definition
High	Open Redirect	OPT.VBNET.OpenRedirect	OpenRedirect: URL Redirection to Untrusted ('Open Redirect')
High	Cross Site History Manipulation	OPT.VBNET.SEC.CrossSiteHistoryManipulation	CrossSiteHistoryManipulation: Cross-Site History Manipulation (XSHM)
High	Log Forging	OPT.VBNET.SEC.LogForging	LogForging: Improper Output Neutralization in Logs
High	Resource Injection	OPT.VBNET.SEC.ResourceInjection	ResourceInjection: Improper control of resource identifiers ("Resource Injection")
High	Trust Boundary Violation	OPT.VBNET.SEC.TrustBoundaryViolation	TrustBoundaryViolation: Trust boundary violation
High	Unsafe Reflection	OPT.VBNET.SEC.UnsafeReflection	UnsafeReflection: Use of Externally-Controlled Input to Select Classes or Code ('Unsafe Reflection')
High	XML Entity Injection	OPT.VBNET.SEC.XMLEntityInjection	XMLEntityInjection: XML entity injection
High	XML Injection	OPT.VBNET.XMLInjection	XMLInjection: XML Injection (aka Blind XPath Injection)
High	XPath Injection	OPT.VBNET.XPathInjection	XPathInjection: Improper Neutralization of DTD within XPath Expressions ('XPath Injection')
High	XQuery Injection	OPT.VBNET.XQueryInjection	XQueryInjection: Improper Neutralization of DTD within XQuery Expressions ('XQuery Injection')
High	XSLT Injection	OPT.VBNET.XSLTInjection	XSLTInjection: Avoid using non-neutralized controlled input when creating XSL stylesheets
High	Information Exposure Through Error Message	OPT.VBNET.SEC.InformationExposureThroughErrorMessage	InformationExposureThroughErrorMessage: sensitive information exposure through error messages
High	Insecure Email Transport	OPT.VBNET.SEC.InsecureEmailTransport	InsecureEmailTransport: Insecure Mail Transport

Severity	Contrast rule	Engine rule ID	Description
High	Review Visible Event Handlers	OPT.VBNET.VBnet.ReviewVisibleEventHandlers	ReviewVisibleEventHandlers: A public or protected event-handling method was detected
High	Resource Leak Database	OPT.VBNET.ResourceLeakDatabase	ResourceLeakDatabase: Unreleased database resource
High	Resource Leak Ldap	OPT.VBNET.ResourceLeakLdap	ResourceLeakLdap: Unreleased LDAP resource
High	Resource Leak Stream	OPT.VBNET.ResourceLeakStream	ResourceLeakStream: Unreleased stream resource
High	Resource Leak Unmanaged	OPT.VBNET.ResourceLeakUnmanaged	ResourceLeakUnmanaged: Unreleased unmanaged resource
High	Avoid Certificate Equals	OPT.VBNET.SEC.AvoidCertificateEquals	AvoidCertificateEquals: Never use X509Certificate.Equals() in a security context
High	Cookies In Security Decision	OPT.VBNET.SEC.CookiesInSecurityDecision	CookiesInSecurityDecision: Reliance on Cookies without Validation and Integrity Checking in Security Decision
High	Improper Authentication	OPT.VBNET.SEC.ImproperAuthentication	ImproperAuthentication: Avoid that a user can perform actions to which he does not have access
High	Missing Standard Error Handling	OPT.VBNET.SEC.MissingStandardErrorHandling	MissingStandardErrorHandling: Missing Standardized Error Handling Mechanism in ASP.Net
High	Setting Manipulation	OPT.VBNET.SEC.SettingManipulation	SettingManipulation: Setting Manipulation
High	Unvalidated Asp Net Model	OPT.VBNET.SEC.UnvalidatedAspNetModel	UnvalidatedAspNetModel: Unvalidated model MVC controller
High	User Controlled SQL Primary Key	OPT.VBNET.SEC.UserControlledSQLPrimaryKey	UserControlledSQLPrimaryKey: Avoid using user controlled Primary Key into a query
High	Abstract Types Should Not Have Constructors	OPT.VBNET.VBnet.AbstractTypesShouldNotHaveConstructors	AbstractTypesShouldNotHaveConstructors: abstract class with public constructor
High	Attribute Usage	OPT.VBNET.VBnet.AttributeUsage	AttributeUsage: Specify AttributeUsage
High	Avoid Excessive Complexity	OPT.VBNET.VBnet.AvoidExcessiveComplexity	AvoidExcessiveComplexity: A method has excessive cyclomatic complexity
High	Avoid Excessive Locals	OPT.VBNET.VBnet.AvoidExcessiveLocals	AvoidExcessiveLocals: Avoid excessive local variables
High	Avoid Floating Point Equality	OPT.VBNET.VBnet.AvoidFloatingPointEquality	AvoidFloatingPointEquality: Do not perform (in)equality operations over floating point values
High	Avoid Inconditional Recursive Invocation	OPT.VBNET.VBnet.AvoidInconditionalRecursiveInvocation	AvoidInconditionalRecursiveInvocation: Avoid recursive calls without a precondition
High	Avoid Out Parameters	OPT.VBNET.VBnet.AvoidOutParameters	AvoidOutParameters: Public or protected method has ByRef parameters
High	Avoid Overloads In Com Visible Interfaces	OPT.VBNET.VBnet.AvoidOverloadsInComVisibleInterfaces	AvoidOverloadsInComVisibleInterfaces: COM visible interface declares overloaded methods
High	Avoid Uncalled Private Code	OPT.VBNET.VBnet.AvoidUncalledPrivateCode	AvoidUncalledPrivateCode: Avoid uncalled private code
High	Avoid Unused Private Fields	OPT.VBNET.VBnet.AvoidUnusedPrivateFields	AvoidUnusedPrivateFields: Avoid unused private fields
High	Declare Event Handlers Correctly	OPT.VBNET.VBnet.DeclareEventHandlersCorrectly	DeclareEventHandlersCorrectly: Declare Event Handlers Correctly

Severity	Contrast rule	Engine rule ID	Description
High	Do Not Assume Int Ptr Size Rule	OPT.VBNET.VBnet.DoNotAssumeIntPtrSizeRule	DoNotAssumeIntPtrSizeRule: Do not downcast IntPtr or UIntPtr into a 32-bit or smaller value
High	Do Not Call Overridable Methods In Constructors	OPT.VBNET.VBnet.DoNotCallOverridableMethodsInConstructors	DoNotCallOverridableMethodsInConstructors: Do not call virtual method call from constructor
High	Do Not Declare Static Members On Generic Types	OPT.VBNET.VBnet.DoNotDeclareStaticMembersOnGenericTypes	DoNotDeclareStaticMembersOnGenericTypes: Do not declare shared members on generic types
High	Do Not Lock On This Or Types	OPT.VBNET.VBnet.DoNotLockOnThisOrTypes	DoNotLockOnThisOrTypes: Do not lock on object or over its 'Type'
High	Do Not Pass Types By Reference	OPT.VBNET.VBnet.DoNotPassTypesByReference	DoNotPassTypesByReference: Do not pass types by reference
High	Empty Catch	OPT.VBNET.VBnet.EmptyCatch	EmptyCatch: Do not leave empty catch blocks
High	Exceptions Should Be Public	OPT.VBNET.VBnet.ExceptionsShouldBePublic	ExceptionsShouldBePublic: Exceptions should be public
High	Get Hash Code Equals	OPT.VBNET.VBnet.GetHashCodeEquals	GetHashCodeEquals: Overload GetHashCode and Equals as a whole
High	Identifiers Should Not Match Keywords	OPT.VBNET.VBnet.IdentifiersShouldNotMatchKeywords	IdentifiersShouldNotMatchKeywords: The identifiers must not be reserved words
High	Initialize Reference Type Static Fields Inline	OPT.VBNET.VBnet.InitializeReferenceTypeStaticFieldsInline	InitializeReferenceTypeStaticFieldsInline: Initialize reference type static fields inline
High	Mark Boolean P Invoke Arguments With Marshal As	OPT.VBNET.VBnet.MarkBooleanPInvokeArgumentsWithMarshalAs	MarkBooleanPInvokeArgumentsWithMarshalAs: Mark boolean P/Invoke arguments with MarshalAs attribute
High	Max Methods	OPT.VBNET.VBnet.MaxMethods	MaxMethods: Maximum allowed number of methods
High	Non Constant Fields Should Not Be Visible	OPT.VBNET.VBnet.NonConstantFieldsShouldNotBeVisible	NonConstantFieldsShouldNotBeVisible: A public or protected static field is not constant nor is read-only
High	Normalize Strings To Uppercase	OPT.VBNET.VBnet.NormalizeStringsToUppercase	NormalizeStringsToUppercase: Not to convert string chains to lower case letters
High	Obsolete Usages	OPT.VBNET.VBnet.ObsoleteUsages	ObsoleteUsages: Avoid use of deprecated methods or obsolete classes depending on chosen .NET SDK
High	Parameter Names Should Not Match Member Names	OPT.VBNET.VBnet.ParameterNamesShouldNotMatchMemberNames	ParameterNamesShouldNotMatchMemberNames: Parameter names should not match member names
High	Properties Should Not Return Arrays	OPT.VBNET.VBnet.PropertiesShouldNotReturnArrays	PropertiesShouldNotReturnArrays: Properties should not return arrays
High	Property Names Should Not Match Get Methods	OPT.VBNET.VBnet.PropertyNamesShouldNotMatchGetMethods	PropertyNamesShouldNotMatchGetMethods: Property names should not match get methods
High	Remove Unused Locals	OPT.VBNET.VBnet.RemoveUnusedLocals	RemoveUnusedLocals: Unused local variables

Severity	Contrast rule	Engine rule ID	Description
High	Insecure Randomness	OPT.VBNET.InsecureRandomness	InsecureRandomness: Standard pseudo-random number generators cannot withstand cryptographic attacks
High	Review Unused Parameters	OPT.VBNET.VBnet.ReviewUnusedParameters	ReviewUnusedParameters: Method parameters not used
High	Hardcoded Crypto Key	OPT.VBNET.SEC.HardcodedCryptoKey	HardcodedCryptoKey: Use of Hard-coded Cryptographic Key
High	Suppress Finalize Correctly	OPT.VBNET.VBnet.SuppressFinalizeCorrectly	SuppressFinalizeCorrectly: Call GC.SuppressFinalize correctly
High	Use Safe Handle To Encapsulate Native Resources	OPT.VBNET.VBnet.UseSafeHandleToEncapsulateNativeResources	UseSafeHandleToEncapsulateNativeResources: Use of System.IntPtr
High	Variable Names Should Not Match Field Names	OPT.VBNET.VBnet.VariableNamesShouldNotMatchFieldNames	VariableNamesShouldNotMatchFieldNames: Variable names should not match field names
High	Hardcoded Salt	OPT.VBNET.SEC.HardcodedSalt	HardcodedSalt: A hardcoded salt can compromise system security
High	Insecure Transport	OPT.VBNET.SEC.InsecureTransport	InsecureTransport: Insecure transport
High	Proper Padding With Public Key Crypto	OPT.VBNET.SEC.ProperPaddingWithPublicKeyCrypto	ProperPaddingWithPublicKeyCrypto: Use of Algorithm without Optimal Asymmetric Encryption Padding (OAEP)
High	Server Insecure Transport	OPT.VBNET.SEC.ServerInsecureTransport	ServerInsecureTransport: Insecure transport [HTTP servers]
High	Weak Encryption	OPT.VBNET.WeakEncryption	WeakEncryption: Insufficient RSA key length
Info	Do Not Initialize Unnecessarily	OPT.VBNET.DoNotInitializeUnnecessarily	DoNotInitializeUnnecessarily: Do not initialize variables unnecessarily
Info	Do Not Prefix Enum Values With Type Name	OPT.VBNET.DoNotPrefixEnumValuesWithTypeName	DoNotPrefixEnumValuesWithTypeName: Names of enumeration members should not start with enumeration name
Info	Avoid Unneeded Calls On String	OPT.VBNET.VBnet.AvoidUnneededCallsOnString	AvoidUnneededCallsOnString: Avoid unnecessary calls on string objects
Info	Do Not Hardcode Locale Specific Strings	OPT.VBNET.VBnet.DoNotHardcodeLocaleSpecificStrings	DoNotHardcodeLocaleSpecificStrings: Do not hardcode locale specific strings
Info	Identifiers Should Have Correct Suffix	OPT.VBNET.VBnet.IdentifiersShouldHaveCorrectSuffix	IdentifiersShouldHaveCorrectSuffix: An identifier does not have the correct suffix
Info	Identifiers Should Not Contain Underscores	OPT.VBNET.VBnet.IdentifiersShouldNotContainUnderscores	IdentifiersShouldNotContainUnderscores: An identifier contains the underscore character
Info	Identifiers Should Not Have Incorrect Suffix	OPT.VBNET.VBnet.IdentifiersShouldNotHaveIncorrectSuffix	IdentifiersShouldNotHaveIncorrectSuffix: An identifier has an incorrect suffix
Info	Interface Naming Conventions	OPT.VBNET.VBnet.InterfaceNamingConventions	InterfaceNamingConventions: Follow naming conventions in interfaces
Info	Naming Conventions	OPT.VBNET.VBnet.NamingConventions	NamingConventions: Follow naming conventions for classes

Severity	Contrast rule	Engine rule ID	Description
Low	Constants Should Be Transparent	OPT.VBNET.ConstantsShouldBeTransparent	ConstantsShouldBeTransparent: A field constant or an enumeration member should be transparent
Low	Information Exposure Through Debug Log	OPT.VBNET.SEC.InformationExposureThroughDebugLog	InformationExposureThroughDebugLog: Avoid exposing sensible information through log
Low	Avoid Language Specific Type Names In Parameters	OPT.VBNET.AvoidLanguageSpecificTypeNamesInParameters	AvoidLanguageSpecificTypeNamesInParameters: Avoid names of parameters that contains language-specific type name, in public methods
Low	Avoid Unnecessary String Creation	OPT.VBNET.AvoidUnnecessaryStringCreation	AvoidUnnecessaryStringCreation: Avoid creating unnecessary strings through a call to System.String.ToLower or System.String.ToUpper
Low	Collection Properties Should Be Read Only	OPT.VBNET.CollectionPropertiesShouldBeReadOnly	CollectionPropertiesShouldBeReadOnly: Public properties that are of a type that implements System.Collections.ICollection should be read-only
Low	Consider Converting Method To Property	OPT.VBNET.ConsiderConvertingMethodToProperty	ConsiderConvertingMethodToProperty: Consider converting the method into a property
Low	Disposable Types Should Declare Finalizer	OPT.VBNET.DisposableTypesShouldDeclareFinalizer	DisposableTypesShouldDeclareFinalizer: A type that implements System.IDisposable and has fields of unmanaged types, should have a finalizer
Low	Do Not Concatenate Strings Inside Loops	OPT.VBNET.DoNotConcatenateStringsInsideLoops	DoNotConcatenateStringsInsideLoops: Do not concatenate string values inside loops
Low	Do Not Mark Enums With Flags	OPT.VBNET.DoNotMarkEnumsWithFlags	DoNotMarkEnumsWithFlags: The values of an enumeration that has System.FlagsAttribute attribute, should be powers of two
Low	Identifiers Should Be Cased Correctly	OPT.VBNET.IdentifiersShouldBeCasedCorrectly	IdentifiersShouldBeCasedCorrectly: Use consistent "pascal-case" and "camel-case" according to the agreement established
Low	Implement Serialization Constructors	OPT.VBNET.ImplementSerializationConstructors	ImplementSerializationConstructors: A type that implements ISerializable, should implement a serialization constructor
Low	Implement Serialization Methods Correctly	OPT.VBNET.ImplementSerializationMethodsCorrectly	ImplementSerializationMethodsCorrectly: Implement a correct signature for methods that handle serialization events
Low	Initialize Value Type Static Fields Inline	OPT.VBNET.InitializeValueTypeStaticFieldsInline	InitializeValueTypeStaticFieldsInline: Avoid explicitly declare a static constructor
Low	Instantiate Argument Exceptions Correctly	OPT.VBNET.InstantiateArgumentExceptionsCorrectly	InstantiateArgumentExceptionsCorrectly: Avoid calling default constructor of ArgumentException
Low	Operator Overloads Have Named Alternates	OPT.VBNET.OperatorOverloadsHaveNamedAlternates	OperatorOverloadsHaveNamedAlternates: When a type overrides an operator, it is recommended to also override the alternative method
Low	Overload Operator Equals On Overriding Equals	OPT.VBNET.OverloadOperatorEqualsOnOverridingEquals	OverloadOperatorEqualsOnOverridingEquals: When a type overrides System.Object.Equals method, it also should override the operator

Severity	Contrast rule	Engine rule ID	Description
Low	Prefer Jagged Arrays Over Multidimensional	OPT.VBNET.PreferJaggedArraysOverMultidimensional	PreferJaggedArraysOverMultidimensional: declare multidimensional array
Low	Use Literals Where Appropriate	OPT.VBNET.UseLiteralsWhereAppropriate	UseLiteralsWhereAppropriate: Do not declare shared and readonly fields, that are initialized with a value
Low	Use Managed Equivalents Of Win32 Api	OPT.VBNET.UseManagedEquivalentsOfWin32Api	UseManagedEquivalentsOfWin32Api: Use alternatives to existing methods for managing win32 api
Low	Attribute String Literals Should Parse Correctly	OPT.VBNET.VBnet.AttributeStringLiteralsShouldParseCorrectly	AttributeStringLiteralsShouldParseCorrectly: Attribute's literal value should be correctly parsed
Low	Avoid Long Parameter Lists	OPT.VBNET.VBnet.AvoidLongParameterLists	AvoidLongParameterLists: Do not encode methods with too many parameters
Low	Avoid Namespaces With Few Types	OPT.VBNET.VBnet.AvoidNamespacesWithFewTypes	AvoidNamespacesWithFewTypes: Avoid namespaces with few types
Low	Avoid Non Stored Procedure Commands	OPT.VBNET.VBnet.AvoidNonStoredProcedureCommands	AvoidNonStoredProcedureCommands: Avoid using non stored-procedure database operations
Low	Avoid Type Get Type For Constant Strings	OPT.VBNET.VBnet.AvoidTypeGetTypeForConstantStrings	AvoidTypeGetTypeForConstantStrings: Do not use Type.GetType() with constant string values
Low	Avoid Uninstantiated Internal Classes	OPT.VBNET.VBnet.AvoidUninstantiatedInternalClasses	AvoidUninstantiatedInternalClasses: An instance of an assembly-level type is not created by the assembly
Low	Call Base Class Methods On I Serializable Types	OPT.VBNET.VBnet.CallBaseClassMethodsOnISerializableTypes	CallBaseClassMethodsOnISerializableTypes: Call base class methods on ISerializable types
Low	Check New Exception Without Throwing	OPT.VBNET.VBnet.CheckNewExceptionWithoutThrowing	CheckNewExceptionWithoutThrowing: Exception instantiation not used
Low	Consider Custom Accessors For Non Visible Events Rule	OPT.VBNET.Vbnet.ConsiderCustomAccessorsForNonVisibleEventsRule	ConsiderCustomAccessorsForNonVisibleEventsRule: For non visible events, evaluate using custom accessors instead of default ones
Low	Delegates Passed To Native Code Must Include Exception Handling Rule	OPT.VBNET.VBnet.DelegatesPassedToNativeCodeMustIncludeExceptionHandlingRule	DelegatesPassedToNativeCodeMustIncludeExceptionHandlingRule: Enclose with a catch all handler the entire block for delegates passed to native code
Low	Do Not Cast Unnecessarily	OPT.VBNET.VBnet.DoNotCastUnnecessarily	DoNotCastUnnecessarily: A method performs unnecessary duplicate casts on one of its arguments or local variables
Low	Do Not Destroy Stack Trace Rule	OPT.VBNET.VBnet.DoNotDestroyStackTraceRule	DoNotDestroyStackTraceRule: Catch handlers should rethrow original exception instead of throwing the same exception
Low	Do Not Indirectly Expose Methods With Link Demands	OPT.VBNET.VBnet.DoNotIndirectlyExposeMethodsWithLinkDemands	DoNotIndirectlyExposeMethodsWithLinkDemands: Do not indirectly expose methods with link demands

Severity	Contrast rule	Engine rule ID	Description
Low	Do Not Raise Exceptions In Unexpected Locations	OPT.VBNET.Vbnet.DoNotRaiseExceptionsInUnexpectedLocations	DoNotRaiseExceptionsInUnexpectedLocations: Do not raise exceptions in unexpected locations
Low	Do Not Use Thread Static With Instance Fields	OPT.VBNET.VBnet.DoNotUseThreadStaticWithInstanceFields	DoNotUseThreadStaticWithInstanceFields: use 'ThreadStatic' with instance fields
Low	Num Max Class By Namespaces	OPT.VBNET.VBnet.NumMaxClassByNamespaces	NumMaxClassByNamespaces: Avoid an excessive number of classes per package/namespace
Low	Only Flags Enums Should Have Plural Names	OPT.VBNET.VBnet.OnlyFlagsEnumsShouldHavePluralNames	OnlyFlagsEnumsShouldHavePluralNames: Externally visible enumeration ends in a plural word and is not marked with the Flags attribute
Low	Operations Should Not Overflow	OPT.VBNET.VBnet.OperationsShouldNotOverflow	OperationsShouldNotOverflow: Operations should not overflow
Low	Override Equals On Value Types	OPT.VBNET.VBnet.OverrideEqualsOnValueTypes	OverrideEqualsOnValueTypes: A public value type does not override Equals
Low	Remove Empty Finalizers	OPT.VBNET.VBnet.RemoveEmptyFinalizers	RemoveEmptyFinalizers: Remove empty finalizers
Low	Test For Empty Strings Using Length	OPT.VBNET.VBnet.TestForEmptyStringsUsingLength	TestForEmptyStringsUsingLength: Compare strings with empty string using 'Equals'
Low	Types That Own Native Resources Should Be Disposable	OPT.VBNET.VBnet.TypesThatOwnNativeResourcesShouldBeDisposable	TypesThatOwnNativeResourcesShouldBeDisposable: Types that own native resources should be disposable
Low	Write Static Field From Instance Method	OPT.VBNET.VBnet.WriteStaticFieldFromInstanceMethod	WriteStaticFieldFromInstanceMethod: Do not write in static fields from instance methods
Medium	Unsafe Cookie Rule	OPT.VBNET.SEC.UnsafeCookieRule	UnsafeCookieRule: Generate server-side cookies with adequate security properties
Medium	Avoid Host Name Checks	OPT.VBNET.SEC.AvoidHostNameChecks	AvoidHostNameChecks: Avoid checks on client-side hostname, that are not reliable due to DNS poisoning
Medium	MVC Remove Version Header	OPT.VBNET.MVCRemoveVersionHeader	MVCRemoveVersionHeader: Remove ASP.NET MVC version from HTTP headers
Medium	P Invokes Should Not Be Safe Critical	OPT.VBNET.PInvokesShouldNotBeSafeCritical	PInvokesShouldNotBeSafeCritical: A P/Invoke declaration should not have the SecuritySafeCriticalAttribute attribute
Medium	Hardcoded Credential	OPT.VBNET.SEC.HardcodedCredential	HardcodedCredential: Use of Hard-coded Credentials
Medium	Hardcoded Network Address	OPT.VBNET.SEC.HardcodedNetworkAddress	HardcodedNetworkAddress: Network addresses should not be hardcoded
Medium	Plaintext Storage Of Password	OPT.VBNET.SEC.PlaintextStorageOfPassword	PlaintextStorageOfPassword: Plaintext Storage of a Password
Medium	Serializable Class Containing Sensitive Data	OPT.VBNET.SEC.SerializableClassContainingSensitiveData	SerializableClassContainingSensitiveData: Serializable Class Containing Sensitive Data
Medium	Secure Serialization Constructors	OPT.VBNET.SecureSerializationConstructors	SecureSerializationConstructors: Serialization constructors should be protected with security demands
Medium	Secured Types Should Not Expose Fields	OPT.VBNET.SecuredTypesShouldNotExposeFields	SecuredTypesShouldNotExposeFields: Types secured with Link Demands should not expose fields

Severity	Contrast rule	Engine rule ID	Description
Medium	Transparency Annotations Should Not Conflict	OPT.VBNET.TransparencyAnnotationsShouldNotConflict	TransparencyAnnotationsShouldNotConflict: The security attribute of a type should have the same transparency that the security attributes of its members that it contains
Medium	Do Not Expose Fields In Secured Type	OPT.VBNET.VBnet.DoNotExposeFieldsInSecuredType	DoNotExposeFieldsInSecuredType: Do not declare public types that are secured but also expose its fields
Medium	Review Suppress Unmanaged Code Security Usage	OPT.VBNET.VBnet.ReviewSuppressUnmanagedCodeSecurityUsage	ReviewSuppressUnmanagedCodeSecurityUsage: Do not use the 'SuppressUnmanagedCodeSecurity' attribute
Medium	Avoid Readonly Mutable Types	OPT.VBNET.AvoidReadonlyMutableTypes	AvoidReadonlyMutableTypes: Do not declare externally visible read-only fields with mutable types
Medium	Call GC Keep Alive When Using a Native Resource	OPT.VBNET.CallGCKeepAliveWhenUsingaNativeResource	CallGCKeepAliveWhenUsingaNativeResource: GC.KeepAlive should be called in methods that use unmanaged resources
Medium	Critical Types Must Not Participate In Type Equivalence	OPT.VBNET.CriticalTypesMustNotParticipateInTypeEquivalence	CriticalTypesMustNotParticipateInTypeEquivalence: SecurityCriticalAttribute should not be used on members, or types that participate in type equivalence
Medium	Dispose Methods Should Call Suppress Finalize	OPT.VBNET.DisposeMethodsShouldCallSuppressFinalize	DisposeMethodsShouldCallSuppressFinalize: Dispose method of a class that implements System.IDisposable, should call GC.SuppressFinalize
Medium	Method Security Should Be Superset Of Type	OPT.VBNET.MethodSecurityShouldBeSupersetOfType	MethodSecurityShouldBeSupersetOfType: Security of methods should be a subset of the security of types
Medium	MVC Post In Controllers	OPT.VBNET.MVCPostInControllers	MVCPostInControllers: Restrict allowed HTTP verbs for state-change operations in MVC controllers
Medium	Potential Infinite Loop	OPT.VBNET.PotentialInfiniteLoop	PotentialInfiniteLoop: Loop with Unreachable Condition ('Infinite Loop')
Medium	Provide Correct Arguments To Formatting Methods	OPT.VBNET.ProvideCorrectArgumentsToFormattingMethods	ProvideCorrectArgumentsToFormattingMethods: The format argument passed to System.String.Format does not match with the objects passed as parameters
Medium	Provide Deserialization Methods For Optional Fields	OPT.VBNET.ProvideDeserializationMethodsForOptionalFields	ProvideDeserializationMethodsForOptionalFields: Provide methods for the de-serialization of fields marked with OptionalFieldAttribute
Medium	Review Declarative Security On Value Types	OPT.VBNET.ReviewDeclarativeSecurityOnValueTypes	ReviewDeclarativeSecurityOnValueTypes: Avoid using declarative security in value types
Medium	Review Imperative Security	OPT.VBNET.ReviewImperativeSecurity	ReviewImperativeSecurity: Avoid using the imperative security whenever possible
Medium	Http Request Value Shadowing	OPT.VBNET.SEC.HttpRequestValueShadowing	HttpRequestValueShadowing: Request data accessed in an ambiguous way, which can be open to attack
Medium	Main Method In Web Application	OPT.VBNET.SEC.MainMethodInWebApplication	MainMethodInWebApplication: Main() method is not allowed in web application
Medium	System Information Leak	OPT.VBNET.SystemInformationLeak	SystemInformationLeak: Exposure of System Data to an Unauthorized Control Sphere

Severity	Contrast rule	Engine rule ID	Description
Medium	Test For NaN Correctly	OPT.VBNET.TestForNaNCorrectly	TestForNaNCorrectly: Not use NaN in the expressions for developing equality test
Medium	Transparent Methods Must Not Call Native Code	OPT.VBNET.TransparentMethodsMustNotCallNativeCode	TransparentMethodsMustNotCallNativeCode: A transparent method should not make calls to native code
Medium	Transparent Methods Must Not Handle Process Corrupting Exceptions	OPT.VBNET.TransparentMethodsMustNotHandleProcessCorruptingExceptions	TransparentMethodsMustNotHandleProcessCorruptingExceptions: A transparent method must not have the attribute HandleProcessCorruptedStateExceptionsAttribute
Medium	Transparent Methods Should Not Be Protected With Link Demands	OPT.VBNET.TransparentMethodsShouldNotBeProtectedWithLinkDemands	TransparentMethodsShouldNotBeProtectedWithLinkDemands: A transparent method should not require LinkDemand
Medium	Transparent Methods Should Not Demand	OPT.VBNET.TransparentMethodsShouldNotDemand	TransparentMethodsShouldNotDemand: A transparent method should not require SecurityAction.Demand, and should not call CodeAccessPermission.Demand method
Medium	Transparent Methods Should Not Load Assemblies From Byte Arrays	OPT.VBNET.TransparentMethodsShouldNotLoadAssembliesFromByteArrays	TransparentMethodsShouldNotLoadAssembliesFromByteArrays: A transparent method should not load an assembly from a byte array using the Assembly.Load method
Medium	Transparent Methods Should Not Use SuppressUnmanagedCodeSecurity	OPT.VBNET.TransparentMethodsShouldNotUseSuppressUnmanagedCodeSecurity	TransparentMethodsShouldNotUseSuppressUnmanagedCodeSecurity: A transparent method should not have the attribute SuppressUnmanagedCodeSecurityAttribute
Medium	Type Link Demands Require Inheritance Demands	OPT.VBNET.TypeLinkDemandsRequireInheritanceDemands	TypeLinkDemandsRequireInheritanceDemands: A public type protected with link demand requires inheritance demand
Medium	Unchecked Input In Loop Condition	OPT.VBNET.UncheckedInputInLoopCondition	UncheckedInputInLoopCondition: Unchecked input in loop condition
Medium	Unchecked Return Value	OPT.VBNET.UncheckedReturnValue	UncheckedReturnValue: Unchecked return value
Medium	Array Fields Should Not Be Read Only	OPT.VBNET.VBnet.ArrayFieldsShouldNotBeReadOnly	ArrayFieldsShouldNotBeReadOnly: Array fields should not be read only
Medium	Attribute Suffix	OPT.VBNET.VBnet.AttributeSuffix	AttributeSuffix: Name of an attribute class should be postfixed with 'Attribute'
Medium	Avoid Calling Problematic Methods	OPT.VBNET.VBnet.AvoidCallingProblematicMethods	AvoidCallingProblematicMethods: Potential dangerous call
Medium	Avoid Large Methods	OPT.VBNET.VBnet.AvoidLargeMethods	AvoidLargeMethods: Avoid functions and methods with too many lines of code
Medium	Avoid Large Structure	OPT.VBNET.VBnet.AvoidLargeStructure	AvoidLargeStructure: Avoid creating structures that are too large
Medium	Avoid Protected Instance Fields	OPT.VBNET.VBnet.AvoidProtectedInstanceFields	AvoidProtectedInstanceFields: Avoid Protected public fields
Medium	Avoid Static Members In Com Visible Types	OPT.VBNET.VBnet.AvoidStaticMembersInComVisibleTypes	AvoidStaticMembersInComVisibleTypes: Avoid static members in COM visible types

Severity	Contrast rule	Engine rule ID	Description
Medium	Avoid Unsealed Concrete Attributes Rule	OPT.VBNET.VBnet.AvoidUnsealedConcreteAttributesRule	AvoidUnsealedConcreteAttributesRule: Avoid attributes defined as NotInheritable but not abstract
Medium	Bad Exception Parent	OPT.VBNET.VBnet.BadExceptionParent	BadExceptionParent: Custom Exception should not derive from certain 'not allowed' base exception classes
Medium	Bad Exception Thrown	OPT.VBNET.VBnet.BadExceptionThrown	BadExceptionThrown: Illegal Exception Thrown. Exceptions must be defined in separated class
Medium	Call Get Last Error Immediately After P Invoke	OPT.VBNET.VBnet.CallGetLastErrorImmediatelyAfterPInvoke	CallGetLastErrorImmediatelyAfterPInvoke: GetLastError immediately after P/Invoke
Medium	Check New Thread Without Start	OPT.VBNET.VBnet.CheckNewThreadWithoutStart	CheckNewThreadWithoutStart: Avoid creating unstarted threads
Medium	Clone Method Should Not Return Null	OPT.VBNET.VBnet.CloneMethodShouldNotReturnNull	CloneMethodShouldNotReturnNull: An override of Clone() method should never return null
Medium	Collection Suffix	OPT.VBNET.VBnet.CollectionSuffix	CollectionSuffix: Classes that implement Collection interfaces should have the 'Collection' suffix
Medium	Collections Should Implement Generic Interface	OPT.VBNET.VBnet.CollectionsShouldImplementGenericInterface	CollectionsShouldImplementGenericInterface: Collections should implement the generic interface
Medium	Com Visible Type Base Types Should Be Com Visible	OPT.VBNET.VBnet.ComVisibleTypeBaseTypesShouldBeComVisible	ComVisibleTypeBaseTypesShouldBeComVisible: COM visible type derive from a non COM visible type
Medium	Common Exception Bases	OPT.VBNET.VBnet.CommonExceptionBases	CommonExceptionBases: Derive Custom Exceptions from allowed classes defined in rule's properties
Medium	Consider Passing Base Types As Parameters	OPT.VBNET.VBnet.ConsiderPassingBaseTypesAsParameters	ConsiderPassingBaseTypesAsParameters: Consider passing base types as parameters
Medium	Declare Types In Namespaces	OPT.VBNET.VBnet.DeclareTypesInNamespaces	DeclareTypesInNamespaces: Declare types in namespaces
Medium	Default Parameters Should Not Be Used	OPT.VBNET.VBnet.DefaultParametersShouldNotBeUsed	DefaultParametersShouldNotBeUsed: Do not use default parameters
Medium	Disable Debugging Code Rule	OPT.VBNET.VBnet.DisableDebuggingCodeRule	DisableDebuggingCodeRule: Avoid using Console.WriteLine
Medium	Disposable Fields Should Be Disposed	OPT.VBNET.VBnet.DisposableFieldsShouldBeDisposed	DisposableFieldsShouldBeDisposed: Call Dispose method of fields that implements IDisposable
Medium	Dispose Finalize	OPT.VBNET.VBnet.DisposeFinalize	DisposeFinalize: Implement both Finalize and Dispose
Medium	Do Not Catch General Exception Types	OPT.VBNET.VBnet.DoNotCatchGeneralExceptionTypes	DoNotCatchGeneralExceptionTypes: Do not catch general exception types
Medium	Do Not Declare Overridable Members In Not Inheritable Types	OPT.VBNET.VBnet.DoNotDeclareOverridableMembersInNotInheritableTypes	DoNotDeclareOverridableMembersInNotInheritableTypes: Do not declare Overridable and not virtual members in NotInheritable classes

Severity	Contrast rule	Engine rule ID	Description
Medium	Do Not Decrease Inherited Member Visibility	OPT.VBNET.VBnet.DoNotDecreaseInheritedMemberVisibility	DoNotDecreaseInheritedMemberVisibility: Do not decrease inherited member visibility
Medium	Do Not Ignore Method Results	OPT.VBNET.VBnet.DoNotIgnoreMethodResults	DoNotIgnoreMethodResults: Do not ignore returning value of methods
Medium	Do Not Pass Literals As Localized Parameters	OPT.VBNET.VBnet.DoNotPassLiteralsAsLocalizedParameters	DoNotPassLiteralsAsLocalizedParameters: literal passes as parameter should be localized
Medium	Do Not Raise Exceptions In Exception Clauses	OPT.VBNET.VBnet.DoNotRaiseExceptionsInExceptionClauses	DoNotRaiseExceptionsInExceptionClauses: exception is thrown from a finally, filter, or finally clause
Medium	Do Not Raise Reserved Exception Types	OPT.VBNET.VBnet.DoNotRaiseReservedExceptionTypes	DoNotRaiseReservedExceptionTypes: Do not raise reserved exception types
Medium	Do Not Use Timers That Prevent Power State Changes	OPT.VBNET.VBnet.DoNotUseTimersThatPreventPowerStateChanges	DoNotUseTimersThatPreventPowerStateChanges: Avoid timers that prevent power state changes
Medium	Double Check Locking Rule	OPT.VBNET.VBnet.DoubleCheckLockingRule	DoubleCheckLockingRule: Incorrect usage of double checking when implementing SingleCheck pattern
Medium	Equal Op With Add Sub	OPT.VBNET.VBnet.EqualOpWithAddSub	EqualOpWithAddSub: Implement +, -, and {}
Medium	Equals Throws Ex	OPT.VBNET.VBnet.EqualsThrowsEx	EqualsThrowsEx: Avoid throwing exception in Equals method
Medium	Exception Constructors	OPT.VBNET.VBnet.ExceptionConstructors	ExceptionConstructors: Custom Exception classes should implement the common Constructors
Medium	Exception Suffix	OPT.VBNET.VBnet.ExceptionSuffix	ExceptionSuffix: Classes that inherits Exception classes should have the 'Exception' suffix
Medium	Get Hash Code Throws Ex	OPT.VBNET.VBnet.GetHashCodeThrowsEx	GetHashCodeThrowsEx: Avoid throwing exceptions when overriding GetHashCode
Medium	I Comparable With Comp Ops	OPT.VBNET.VBnet.IComparableWithCompOps	IComparableWithCompOps: Implement comparison operators when implementing IComparable
Medium	Implement I Serializable Correctly	OPT.VBNET.VBnet.ImplementISerializableCorrectly	ImplementISerializableCorrectly: Implement ISerializable correctly
Medium	Implement Standard Exception Constructors	OPT.VBNET.VBnet.ImplementStandardExceptionConstructors	ImplementStandardExceptionConstructors: Implement standard exception constructors
Medium	Index With ICollection	OPT.VBNET.VBnet.IndexWithICollection	IndexWithICollection: Avoid indexed properties on classes not extending from System.Collections.Interfaces
Medium	Level2 Assemblies Should Not Contain Linkdemands	OPT.VBNET.VBnet.Level2AssembliesShouldNotContainLinkdemands	Level2AssembliesShouldNotContainLinkdemands: A class or class member is using a LinkDemands in an application that is using Level 2 security
Medium	Mark Members As Static	OPT.VBNET.VBnet.MarkMembersAsStatic	MarkMembersAsStatic: A method that only accesses class members should be marked 'Shared'

Severity	Contrast rule	Engine rule ID	Description
Medium	Members Should Not Expose Certain Concrete Types	OPT.VBNET.VBnet.MembersShouldNotExposeCertainConcreteTypes	MembersShouldNotExposeCertainConcreteTypes: Members should not expose certain concrete types
Medium	Move P Invokes To Native Methods Class	OPT.VBNET.VBnet.MovePInvokesToNativeMethodsClass	MovePInvokesToNativeMethodsClass: MovePInvokes to NativeMethods class
Medium	Nested Types Should Not Be Visible	OPT.VBNET.VBnet.NestedTypesShouldNotBeVisible	NestedTypesShouldNotBeVisible: An externally visible type contains an externally visible type declaration
Medium	Operator Throws Ex	OPT.VBNET.VBnet.OperatorThrowsEx	OperatorThrowsEx: Overloading a binary operator should not throw exceptions
Medium	Pass System Obj Instead Of String	OPT.VBNET.VBnet.PassSystemObjInsteadOfString	PassSystemObjInsteadOfString: Method invocation can be improved
Medium	Pointers Should Not Be Visible	OPT.VBNET.VBnet.PointersShouldNotBeVisible	PointersShouldNotBeVisible: Pointers should not be visible
Medium	Properties Matched By Constructor Args	OPT.VBNET.VBnet.PropertiesMatchedByConstructorArgs	PropertiesMatchedByConstructorArgs: Constructor parameters set to Properties
Medium	Properties Should Not Be Write Only	OPT.VBNET.VBnet.PropertiesShouldNotBeWriteOnly	PropertiesShouldNotBeWriteOnly: Avoid write-only properties
Medium	Rethrow To Preserve Stack Details	OPT.VBNET.VBnet.RethrowToPreserveStackDetails	RethrowToPreserveStackDetails: Do not rethrow exceptions explicitly
Medium	Same Namespace Type	OPT.VBNET.VBnet.SameNamespaceType	SameNamespaceType: Conflict between namespace and class names
Medium	Set Locale For Data Types	OPT.VBNET.VBnet.SetLocaleForDataTypes	SetLocaleForDataTypes: Set locale property for data types
Medium	Specify Culture Info	OPT.VBNET.VBnet.SpecifyCultureInfo	SpecifyCultureInfo: Specify CultureInfo
Medium	Specify Message Box Options	OPT.VBNET.VBnet.SpecifyMessageBoxOptions	SpecifyMessageBoxOptions: Specify MessageBoxOptions
Medium	Specify String Comparison	OPT.VBNET.VBnet.SpecifyStringComparison	SpecifyStringComparison: Specify StringComparison
Medium	Static Holder Types Should Be Sealed	OPT.VBNET.VBnet.StaticHolderTypesShouldBeSealed	StaticHolderTypesShouldBeSealed: Class containing only static members should be declared NotInheritable
Medium	Static Holder Types Should Not Have Constructors	OPT.VBNET.VBnet.StaticHolderTypesShouldNotHaveConstructors	StaticHolderTypesShouldNotHaveConstructors: Static holder types should not have constructors
Medium	Struct Empty Constructor	OPT.VBNET.VBnet.StructEmptyConstructor	StructEmptyConstructor: Avoid Structures with empty constructors
Medium	Type Names Should Not Match Namespaces	OPT.VBNET.VBnet.TypeNamesShouldNotMatchNamespaces	TypeNamesShouldNotMatchNamespaces: Type names should not match namespaces
Medium	Types Should Not Extend Certain Base Types	OPT.VBNET.VBnet.TypesShouldNotExtendCertainBaseTypes	TypesShouldNotExtendCertainBaseTypes: Types should not extend certain base types
Medium	Uri Parameters Should Not Be Strings	OPT.VBNET.VBnet.UriParametersShouldNotBeStrings	UriParametersShouldNotBeStrings: URI parameters should not be strings

Severity	Contrast rule	Engine rule ID	Description
Medium	Uri Return Values Should Not Be Strings	OPT.VBNET.VBnet.UriReturnValuesShouldNotBeStrings	UriReturnValuesShouldNotBeStrings: The name of a method contains 'uri', 'Uri', 'urn', 'Urn', 'url' or 'Url', and returns a string
Medium	Use Generic Event Handler Instances	OPT.VBNET.VBnet.UseGenericEventHandlerInstances	UseGenericEventHandlerInstances: Use generic event handler instances
Medium	Validate Arguments Of Public Methods	OPT.VBNET.VBnet.ValidateArgumentsOfPublicMethods	ValidateArgumentsOfPublicMethods: Check parameters of externally visible methods

Visual Basic 6 Scan rules

Contrast supports these rules for Visual Basic 6.

Severity	Contrast rule	Engine rule ID	Description
Critical	B I T N	OPT.VB6.VBCC.BITN	BITN: Nested if...then sentences are not allowed
Critical	R A D L	OPT.VB6.VBDS.RADL	RADL: Avoid absolute paths in Lib definition
Critical	O E R N	OPT.VB6.VBEH.OERN	OERN: Use of 'On Error Resume Next'
Critical	K C R A	OPT.VB6.VBFA.KCRA	KCRA: 'Kill' command must not contain absolute paths
Critical	O C R A	OPT.VB6.VBFA.OCRA	OCRA: 'Open' command must not contain absolute paths
Critical	S C R A	OPT.VB6.VBFA.SCRA	SCRA: 'Shell' command must not contain absolute paths
Critical	N C P F	OPT.VB6.VBFD.NCPF	NCPF: Maximum number of controls per form
Critical	F A O E	OPT.VB6.VBSF.FAOE	FAOE: In every source code file must appear 'Option Explicit' declaration
Critical	F R D P	OPT.VB6.VBSF.FRDP	FRDP: Source files must be in the same directory than project (.vpb file)
Critical	E U F P	OPT.VB6.VBUK.EUFP	EUFP: Avoid using pointer functions
Critical	V N U C	OPT.VB6.VBVD.VNUC	VNUC: Public variables must not be used in class modules
High	A N E M	OPT.VB6.VBCC.ANEM	ANEM: Check number of arguments in the methods
High	C P C L	OPT.VB6.VBCC.CPCL	CPCL: Cyclomatic complexity for VB6 procedures should be low
High	D E V L	OPT.VB6.VBCC.DEVL	DEVL: Avoid excessive local variables
High	E U V G	OPT.VB6.VBCC.EUVG	EUVG: Avoid using global variables (except with 'WithEvents')
High	S S D O	OPT.VB6.VBCC.SSDO	SSDO: Do not use nested case statements
High	A C G E	OPT.VB6.VBCD.ACGE	ACGE: Global constants should be Private.
High	E C M N	OPT.VB6.VBCD.ECMN	ECMN: Avoid constants with the same name
High	N C C F	OPT.VB6.VBCD.NCCF	NCCF: Constants must follow a naming standard
High	D C A V	OPT.VB6.VBCL.DCAV	DCAV: Constants declaration should be placed before variables declaration
High	D V C P	OPT.VB6.VBCL.DVCP	DVCP: Variables declaration should be in the headers of the procedures
High	E D L C	OPT.VB6.VBCL.EDLC	EDLC: Avoid Functions and Procedures with more than 200 code lines
High	F P V D	OPT.VB6.VBCL.FPVD	FPVD: Empty function or Procedure detected
High	N E C F	OPT.VB6.VBCL.NECF	NECF: Line label names must be properly formed
High	P E M L	OPT.VB6.VBCL.PEML	PEML: PROPERTY procedures must not exceed certain number of code lines
High	U L I T	OPT.VB6.VBCL.ULIT	ULIT: Do not use inline If...Then statements
High	C G N U	OPT.VB6.VBDC.CGNU	CGNU: Avoid unused Global Constants
High	V G N U	OPT.VB6.VBDC.VGNU	VGNU: Avoid unused Global variables

Severity	Contrast rule	Engine rule ID	Description
High	V L S U	OPT.VB6.VBDC.VLSU	VLSU: Avoid unused Local variables
High	S D A P	OPT.VB6.VBDS.SDAP	SDAP: Declare sentences must have 'Private' scope
High	D E D E	OPT.VB6.VBED.DEDE	DEDE: 'Enum' declaration scope must be 'Public' or 'Private'
High	U E C T	OPT.VB6.VBEH.UECT	UECT: Use 'On Error Resume Next' with 'Class_Terminate' event
High	U O E H	OPT.VB6.VBEH.UOEH	UOEH: Use error-handling routines
High	Number used to describe files (Close)	OPT.VB6.VBFA.CUNF	CUNF: With 'Close' command, numbers must not be used to describe files
High	Number used to describe files (EOF)	OPT.VB6.VBFA.EUNF	EUNF: With 'EOF' command, number must not be used to describe files
High	Number used to describe files (LOF)	OPT.VB6.VBFA.LUNF	LUNF: With 'LOF' command, number must not be used to describe files
High	Number used to describe files (File Access)	OPT.VB6.VBFA.NCAF	NCAF: With file access commands, number must not be used to describe files
High	Close command used, without file number	OPT.VB6.VBFA.NCSF	NCSF: Do not use 'Close' command without file number
High	Number used to describe files (Open)	OPT.VB6.VBFA.OUNF	OUNF: With 'Open' command, number must not be used to describe files
High	B B I T	OPT.VB6.VBFD.BBIT	BBIT: 'Browser' buttons (...) or buttons with only an image should have activated 'ToolTipText' property
High	F C C B	OPT.VB6.VBFD.FCCB	FCCB: Forms must have 'ControlBox'
High	F I E U	OPT.VB6.VBFD.FIEU	FIEU: Forms must implement 'Unload' event
High	F T F P	OPT.VB6.VBFD.FTFP	FTFP: Forms must have the same font type
High	T B M L	OPT.VB6.VBFD.TBML	TBML: MaxLenght property must be specificated
High	I E O M	OPT.VB6.VBOU.IEOM	IEOM: Avoid using 'InStr()' command embedded in 'Mid()' command
High	U O C C	OPT.VB6.VBP.UOCC	UOCC: Avoid using '+' operator to concatenate strings
High	A D P E	OPT.VB6.VBPD.ADPE	ADPE: Procedure declaration scope must be always specified
High	A D T A	OPT.VB6.VBPD.ADTA	ADTA: Formal parameter types must be specified
High	A E V R	OPT.VB6.VBPD.AEVR	AEVR: Formal parameters must be specified as 'ByVal' or 'ByRef'
High	M U P S	OPT.VB6.VBPD.MUPS	MUPS: Methods must have an only exit point
High	N A C F	OPT.VB6.VBPD.NACF	NACF: Formal parameter names must be properly formed
High	O A D V	OPT.VB6.VBPD.OADV	OADV: Optional formal parameters must have a default value
High	N M C P	OPT.VB6.VBSF.NMCP	NMCP: Module names (.bas files) must have a prefix
High	P C N M	OPT.VB6.VBSF.PCNM	PCNM: Class modules names (.cls file) must have a prefix
High	P F M D	OPT.VB6.VBSF.PFMD	PFMD: MDI Forms must have a prefix
High	P F N M	OPT.VB6.VBSF.PFNM	PFNM: No MDI forms must have a prefix
High	P N D F	OPT.VB6.VBSF.PNDF	PNDF: Designer file names (.dsr files) must have a prefix
High	P N P P	OPT.VB6.VBSF.PNPP	PNPP: Property Page file names (.pag files) must have a prefix
High	P N U C	OPT.VB6.VBSF.PNUC	PNUC: User Control file names (.ctl files) must have a prefix
High	P N U D	OPT.VB6.VBSF.PNUD	PNUD: User Document file names (.dob files) must have a prefix
High	E T C N	OPT.VB6.VBTU.ETCN	ETCN: The names of the Type's elements must be properly formed
High	N T C D	OPT.VB6.VBTU.NTCD	NTCD: Type names must follow a naming standard
High	E S O B	OPT.VB6.VBUK.ESOB	ESOB: Do not use 'Option Base' sentence
High	E U F I	OPT.VB6.VBUK.EUFI	EUFI: Avoid using 'IsMissing()' function
High	E V M N	OPT.VB6.VBVD.EVMN	EVMN: Avoid variables with the same name

Severity	Contrast rule	Engine rule ID	Description
High	N V C F	OPT.VB6.VBVD.NVCF	NVCF: Public vars must follow a naming standard
High	N V L F	OPT.VB6.VBVD.NVLF	NVLF: Name of a local variable must follow a naming standard
High	N V S F	OPT.VB6.VBVD.NVSF	NVSF: Static (shared) vars must follow a naming standard
High	N V W F	OPT.VB6.VBVD.NVWF	NVWF: Names of 'WithEvents' variables must be properly formed
High	V O L D	OPT.VB6.VBVD.VOLD	VOLD: Local Object variables must be destroyed
Info	O E G 0	OPT.VB6.VBEH.OEG0	OEG0: Use of 'On Error GoTo 0'
Info	U P A A	OPT.VB6.VBPD.UPAA	UPAA: Avoid using 'ParamArray' in formal parameters
Low	M C P L	OPT.VB6.VBCL.MCPL	MCPL: Maximum number of characters per line
Low	A O L E	OPT.VB6.VBEH.AOLE	AOLE: Avoid using 'On Local Error'
Low	C T F P	OPT.VB6.VBFD.CTFP	CTFP: Controls must have the same font type
Low	N C C P	OPT.VB6.VBFD.NCCP	NCCP: Form control names must be properly formed
Low	A I V N	OPT.VB6.VBPP.AIVN	AIVN: Auto-increment version number must be disabled
Low	V I C N	OPT.VB6.VBPP.VICN	VICN: 'VersionCompanyName' must be defined
Low	V I F D	OPT.VB6.VBPP.VIFD	VIFD: 'VersionFileDescription' must be defined
Low	V I L C	OPT.VB6.VBPP.VILC	VILC: 'VersionLegalCopyright' must be defined
Low	V I P N	OPT.VB6.VBPP.VIPN	VIPN: 'VersionProductName' must be defined
Low	U D M L	OPT.VB6.VBVD.UDML	UDML: Avoid using 'Dim' in global variable declarations
Medium	T C D A	OPT.VB6.VBCD.TCDA	TCDA: Constants must have a defined type
Medium	A G L S	OPT.VB6.VBCL.AGLS	AGLS: Place property Let/Get/Set accessors together
Medium	E S D F	OPT.VB6.VBCL.ESDF	ESDF: Avoid modules that contain only declarations
Medium	C S U	OPT.VB6.VBDC.CSU	CSU: Local constants not used
Medium	P S U	OPT.VB6.VBDC.PSU	PSU: Parameters not used
Medium	E E N P	OPT.VB6.VBED.EENP	EENP: Elements names of a statement Enum must follow a naming standard
Medium	V N E E	OPT.VB6.VBED.VNEE	VNEE: Assign numeric values to 'Enum' elements
Medium	T B N E	OPT.VB6.VBFD.TBNE	TBNE: CommandButton size must be standar
Medium	A N S C	OPT.VB6.VBP.ANSC	ANSC: Avoid String comparisons with empty strings or null string
Medium	A V D V	OPT.VB6.VBP.AVDV	AVDV: Avoid Variant variables
Medium	E T R E	OPT.VB6.VBP.ETRE	ETRE: Specify return type in functions
Medium	U I S C	OPT.VB6.VBP.UISC	UISC: Use String Constants instead of 'Chr\$()' and 'Chr()' functions
Medium	U S C F	OPT.VB6.VBP.USCF	USCF: Use Chr\$() instead of Chr() for Variant variables
Medium	U S D F	OPT.VB6.VBP.USDF	USDF: Use 'CurDir\$()' and 'Dir\$()' functions instead of 'CurDir()' and 'Dir()'
Medium	U S E F	OPT.VB6.VBP.USEF	USEF: Use 'Left\$()', 'Right\$()' and 'Mid\$()' functions instead of 'Left()', 'Right()' and 'Mid()'
Medium	U S H F	OPT.VB6.VBP.USHF	USHF: Use 'Hex\$()' function instead of 'Hex()'
Medium	U S S F	OPT.VB6.VBP.USSF	USSF: Use 'Space\$()' function instead of 'Space()'
Medium	U S T F	OPT.VB6.VBP.USTF	USTF: Use 'LTrim\$()', 'RTrim\$()' and 'Trim\$()' functions instead of 'LTrim()', 'RTrim()' and 'Trim()'
Medium	M I V D	OPT.VB6.VBVD.MIVD	MIVD: Avoid multiple variable declaration in the same declaration statement

XML Scan rules

Contrast Scan supports these rules for XML.

Severity	Contrast rule	Engine rule ID	Description
Critical	Check Action Mappings Type	OPT.XML.STRUTSCONFIG.CheckActionMappingsType	CheckActionMappingsType: Class for action-mappings property does not match with the ones defined in this rule's property
Critical	Check Action With Path Attribute	OPT.XML.STRUTSCONFIG.CheckActionWithPathAttribute	CheckActionWithPathAttribute: Every action must contain a path attribute
Critical	Check Html Redirect Links	OPT.XML.STRUTSCONFIG.CheckHtmlRedirectLinks	CheckHtmlRedirectLinks: A forward to a web document must use the redirect attribute
Critical	Check Maximum Session Scopes	OPT.XML.STRUTSCONFIG.CheckMaximumSessionScopes	CheckMaximumSessionScopes: Level of ActionForm in session overpassed
Critical	Check Name Attribute In Form Beans	OPT.XML.STRUTSCONFIG.CheckNameAttributeInFormBeans	CheckNameAttributeInFormBeans: Every <form-bean> must specify a "name" attribute
Critical	Specify Filter Action	OPT.XML.WEB.SpecifyFilterAction	SpecifyFilterAction: Action filter is not properly defined
Critical	Use the proper slash character in URLs	OPT.XML.XMLPT.USEOFCORRECTBARS	USEOFCORRECTBARS: Use the proper slash character in URLs ('/')
Critical	Document your code	OPT.XML.XSLT_MAN.DOCUMENTEDCODE	DOCUMENTEDCODE: Checks if the code is documented
Critical	Use xsl:choose correctly	OPT.XML.XSLT_MAN.EFFICIENTUSEOFCHOOSE	EFFICIENTUSEOFCHOOSE: Checks if the xsl:choose tag is being used correctly
Critical	Remove unused parameters	OPT.XML.XSLT_MAN.NOUSEDPARAM	NOUSEDPARAM: Detects declared parameters are not being used
Critical	Remove unused variables	OPT.XML.XSLT_MAN.NOUSEDVARIABLES	NOUSEDVARIABLES: Detects declared variables are not being used
Critical	Avoid using axis	OPT.XML.XSLT_OYR.INEFFICIENTAXES	INEFFICIENTAXES: Avoid using axis
Critical	Avoid using XPath comparisons	OPT.XML.XSLT_OYR.NOUSEXPATHEXPRESSIONS	NOUSEXPATHEXPRESSIONS: Advises avoid the direct comparisons between nodes
Critical	Advised the use keys	OPT.XML.XSLT_OYR.USEKEYS	USEKEYS: Advised the use of keys
Critical	Check XPath expressions	OPT.XML.XSLT_PB.CHECKXPATHEXPRESSIONS	CHECKXPATHEXPRESSIONS: Checks the XPath expressions
Critical	Checks tag names xsl:template	OPT.XML.XSLT_PB.DUPLICATENAMEOFTEMPLATE	DUPLICATENAMEOFTEMPLATE: Checks tag names xsl:template
Critical	Checks if the stylesheet is portable	OPT.XML.XSLT_PT.PORTABILITY	PORTABILITY: Checks if the stylesheet is portable
High	Check Unused Action Forms	OPT.XML.STRUTSCONFIG.CheckUnusedActionForms	CheckUnusedActionForms: There are ActionForms that are not being used in actions
Info	Check Form Properties	OPT.XML.STRUTSCONFIG.CheckFormProperties	CheckFormProperties: There are Forms without properties
Info	Compulsory Resources Import	OPT.XML.STRUTSCONFIG.CompulsoryResourcesImport	CompulsoryResourcesImport: Parameter attribute in message-resources is not valid
Medium	Check Action With Forward	OPT.XML.STRUTSCONFIG.CheckActionWithForward	CheckActionWithForward: Every action must be followed by a forward child node

Severity	Contrast rule	Engine rule ID	Description
Medium	Non Matching Type In Form Bean	OPT.XML.STRUTSCONFIG.NonMatchingTypeInFormBean	NonMatchingTypeInFormBean: ActionForm different from DynaValidatorForm or ValidatorForm

Contrast Scan local engine supported versions

Use this topic to understand the supported versions for the Contrast Scan local engine. This information can help you to determine:

- Which versions of the Scan local engine to use.
- How often you need to update.
- Whether you're using versions in your pipelines that Contrast no longer supports.

Local engine versioning

The version number for the Scan local engine comprises three numbers, X.Y.Z (for example 1.1.2), which follows industry standard versioning:

- **X:** A major version of the Scan local engine.
Typically, this version number changes only when the Scan local engine has significant changes. For example, releasing a new engine that fundamentally changes the core architecture represents a major version change.
- **Y:** A minor version of the Scan local engine.
This version number changes for improvements of core components within the Scan local engine. For example, upgrading an existing engine or updating a supported version of Java that the engine uses represents a minor version change.
- **Z:** A maintenance release of the Scan local engine.
Usually, updates for the Scan local engine occur on a monthly basis, therefore, this version number changes with each monthly release.

Scan local engine release notes

To find the latest version of the Scan local engine, visit the [release notes](#). (page 634) The release notes also provide an overview of recent updates.

Support status based on versions

The following table provides guidelines for determining the supported versions of the Scan local engine.

Version number	Supported status
Major (for example, 1.Y.Z)	Current major version only. Where possible, Contrast provides a three-month notice before a Major release is due.
Minor (for example, X.2.Z)	Current — one version (where applicable)
Maintenance (for example: (X.Y.2)	All maintenance versions within the current minor version unless otherwise stated

For example, if the current Scan local engine is version 1.5.8, then unless stated otherwise in the release notes, all versions from 1.4.X onward are considered supported versions.

However if the next release is 2.1.1 then all previous versions of the Scan local engine are considered unsupported. Contrast notifies customers to update their software as soon as possible. Contrast also notifies customers about the new major version three months before the release.

Download the Scan local engine application

Contrast provides a reusable script that lets you download the the latest version of Scan local engine application. The downloaded application is a Java JAR file.

Before you begin

- The script provided in this topic is designed to run in a terminal window as a bash script.
- The script uses these [environment variables \(page 877\)](#):

- CONTRAST__API__ORGANIZATION
- CONTRAST__API__URL
- CONTRAST__API__USER_NAME
- CONTRAST__API__API_KEY
- CONTRAST__API__SERVICE_KEY

To find these keys in the Contrast web interface, select **usermenu > User settings > Profile** and locate the keys under **Your keys**.

Steps

1. Create a script named `download-release.sh` that includes the following code:

```
#!/bin/bash

RELEASE=latest

if [ -n "$1" ]
then
    RELEASE=$1
fi

OUTPUT_FILE=sast-local-scanner-$RELEASE.zip
AUTH_TOKEN=$(echo -n
$CONTRAST__API__USER_NAME:$CONTRAST__API__SERVICE_KEY | base64)

curl \
  -H "api-key: $CONTRAST__API__API_KEY" \
  -H "authorization: $AUTH_TOKEN" \
  -L \
  -o $OUTPUT_FILE \
  $CONTRAST__API__URL/organizations/$CONTRAST__API__ORGANIZATION/release-
artifacts/local-scanner/$RELEASE?download=true
```

2. In a terminal window, set the environment variables with commands similar to the following:

```
export CONTRAST__API__ORGANIZATION=<Contrast_organization_ID>
export CONTRAST__API__URL=https://<your_teamserver_environment>/
Contrast/api/sast
export CONTRAST__API__USER_NAME=<Contrast_user_name>
export CONTRAST__API__API_KEY=<Contrast_API_key>
export CONTRAST__API__SERVICE_KEY=<Contrast_service_key>
```

- Replace `<Contrast_organization_ID>` with your organization ID.
 - Replace `<your_teamserver_environment>` with the address of the Contrast installation where you want to report scan results. For example: `https://teamserver-mycompany/Contrast/api/sast`.
 - Replace `<Contrast_user_name>` with the user name for your Contrast account (usually, your login ID).
 - Replace `<Contrast_API_key>` with your Contrast API key.
 - Replace `<Contrast_service_key>` with your Contrast service key.
3. Run the script using `bash` (for example, `bash download-release.sh`)

You'll see output results similar to the following:

```
janedoe@JDOE-LOCAL Webapps % bash ./download-release.sh
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left     Speed
 0     0    0     0    0     0      0     0  --:--:--  0:00:03  --:--:--    0
82  138M  82  114M    0     0  4389k    0  0:00:32  0:00:26  0:00:06 5011k
```

The script downloads `sast-local-scanner-latest.zip` which includes the latest version of the Scan local engine. The ZIP file is approximately 145 MB in size.

 release-notes.md	21 Mar 2024 at 10:22	222 bytes	Document
 sast-local-scan-runner-1.0.9.jar	21 Mar 2024 at 10:22	154.5 MB	Java JAR file
 sast-local-scan-runner-1.0.9.md5	21 Mar 2024 at 10:22	32 bytes	Document
 sast-local-scan-runner-1.0.9.sha1	21 Mar 2024 at 10:22	40 bytes	Document
 sast-local-scan-runner-1.0.9.sha256	21 Mar 2024 at 10:22	64 bytes	Document

4. Extract the Scan local engine files to a convenient location.
5. Update any custom scripts with the correct version of the Scan local engine

See also

[Run local scan \(page 877\)](#)

Local scan engine environment variables

Use these environment variables with the local Scan engine:

Variable	Required?	Description
CONTRAST__API__URL	Yes	The address of the Contrast installation where you want to report scan results. The URL should include <code>/api/sast</code> at the end of the URL.
CONTRAST__API__USER_NAME	Yes	User name for the Contrast user account (in most cases, your login ID)
CONTRAST__API__API_KEY	Yes	Contrast API key
CONTRAST__API__SERVICE_KEY	Yes	Contrast service key
CONTRAST__API__ORGANIZATION	Yes	Contrast organization ID
CONTRAST__API__PROXY__ENABLE	No	Enables proxy settings
CONTRAST__API__PROXY__URL	Yes, if proxy settings are enabled	The URL for the proxy server (for example, <code>http://host:port</code>)
CONTRAST__API__PROXY__TYPE	Yes, if proxy settings are enabled	The proxy server type (for example, BASIC)
CONTRAST__API__PROXY__USERNAME	No	Username for the proxy server
CONTRAST__API__PROXY__PASSWORD	No	Password for the proxy server

Run local scan

The Contrast Scan local engine is available as a Java JAR file. It requires the location of a build artifact so it can process the scan.

Before you begin

- Identify where the build artifacts that you want to scan are located.
- Determine where you want to store scan results on your local system.
If you don't specify a path for output results, the Scan local engine writes results to the current working directory in a file named `results.sarif`.
- Ensure that this software is installed on your system:
 - **Multi-language source code scans:** Java 17

- **Java binary scans:** Up to Java 17
- **Scans for JavaScript project files:** Semgrep App version 0.114.0
- Internet access exists to let the Scan local engine upload scan results and scanner output to Contrast.
- Verify that you have 1 CPU available (the Contrast Scan local engine is single threaded) and have 12 GB of RAM free.
- Ensure the Scan local engine has read and write permissions for the directory where it runs or for the specified output directory.

**IMPORTANT**

Make sure that the path for the files you want to scan does not include spaces.

Steps

1. Log in to the Contrast web interface.
2. Under the user menu, select **User settings**.
3. In Profile, under Your keys, get the following information:
 - Organization ID
 - Your API key
 - Service key
 - Contrast URL

4. Configure the environment variables that let the local scanner communicate with Contrast:



NOTE

If you want to use a proxy server for communication between the local scan engine and the Contrast platform, include the [proxy server environment variables](#). (page 672)

```
export CONTRAST__API__URL=<URL>
export CONTRAST__API__USER_NAME=<Username>
export CONTRAST__API__API_KEY=<APIKey>
export CONTRAST__API__SERVICE_KEY=<ServiceKey>
export CONTRAST__API__ORGANIZATION=<OrgId>
export LOCAL_ARTIFACT_LOCATION=<LocalArtifactLocation>
export LOCAL_OUTPUT_LOCATION=<LocalOutputLocation>
```

- Replace <URL> with the address of the Contrast installation where you want to report scan results. The URL should include Contrast/api/sast at the end of the URL. For example:

```
export CONTRAST__API__URL=https://app.contrastsecurity.com/
Contrast/api/sast
```

- Replace `<Username>` with your Contrast user account (in most cases your login ID).
- Replace `<APIKey>` with the Contrast API key.
- Replace `<ServiceKey>` with the Contrast service key.
- Replace `<OrgID>` with the Contrast organization ID.
- Replace `<LocalArtifactLocation>` with the path for the JAR or WAR files for Java scans. For JavaScript scans, specify the folder containing the source code.
Optional: You can specify the path in the command instead of using a variable.
- Replace `<LocalOutputLocation>` with the path on the local system where you want to store results files.
Optional: You can specify the path in the command instead of using a variable.

5. Start the scan with a command similar to this one:

```
java -jar sast-local-scan-runner.jar <ScanArtifact> --project- \
name <ProjectName> --label <LabelName>
```

- Replace `<ScanArtifact>` with the path of the JAR, WAR, or ZIP file (which contains only JAR or WAR files) that you want to scan.
For source code scanning, specify the path of the folder that contains the source code you want to scan (not a ZIP file).
For Unix systems, a file name without a path requires `./` in front of the filename.
- Replace `<ProjectName>` with a name for a project. For example: "my project name".
- Replace `<LabelName>` with a label for the scan. For example, you could specify a build number as: "build:1.0.1".

6. Wait several minutes after the scan completes to view results in the Contrast web application. Results are not immediately viewable due to upload and processing times.

Scan branches in repositories

To scan branches in a repository (for example, for a pull request (PR), use the `--branch` option when you run the scan. For example:

```
java -jar sast-local-scan-runner-1.0.10.jar --project-name <ProjectName> --
label <LabelName> --branch <BranchName> <ScanArtifact>
```

Scans of branches return a pass or fail status in the GitHub action. [View details of scanned branches \(page 892\)](#) describes how to view the results in the Contrast web interface.

To configure the scans to fail for new vulnerabilities only, use both the `--fail` and `--branch` options. The scan continues if it finds open vulnerabilities.

If you want to scan to fail when it finds open vulnerabilities use the `--new` option with a value of `false`.

Command options

Use any of these command options:

Option	Description
<code>-o, --output-results</code>	Specifies the location for output results. If you don't specify a location, the local scanner writes the results to the current working directory.
<code>-V, --version</code>	Prints version information
<code>-b, --branch</code>	Specifies a branch in a repository to be scanned. When specified, scan results are aggregated against results for the current branch, without affecting scans for a main branch.

Option	Description
<code>-f, --fail</code>	<p>Fails when the Scan finds open vulnerabilities on a main branch.</p> <p>Branch scanning: If you also specify the <code>--branch</code> option, the scan fails only if it finds new vulnerabilities.</p>
<code>--label <label></code>	A label for the current scan.
<code>--level <level></code>	<p>Turns on logging for the specified log level. Values for this option are:</p> <ul style="list-style-type: none"> • ERROR • WARN • INFO • DEBUG • TRACE <p>Use this option only if Contrast Support instructs you to do so. Don't use it for all scans.</p>
<code>--memory <value></code>	<p>Lets you override the default memory usage of 2 GB.</p> <p>This option is only available for the multi-language source code scan engine.</p>
<code>--metadata <Metatadata> : <Value></code>	<p>Lets you specify metadata as a key-value pair when you create a scan project.</p> <p>If metadata is required, project creation fails if you don't specify the metadata.</p>
<code>-n <true false>, --new <true false></code>	<p>Explicitly specifies whether the scan should fail if it finds new vulnerabilities. The default value is <code>true</code>.</p> <p>Branch scanning: If set to <code>false</code>, the scan fails when it finds open vulnerabilities.</p>
<code>--project-name <ProjectName></code>	<p>The name of a scan project.</p> <p>If you specify a project name that already exists, the Scan local engine adds the scan to that project. Otherwise, it creates a new project with the specified name.</p> <p>If the project name includes spaces, enclose the name in double quotes ("). For example: "My Scan Project".</p> <p>This option is required if you don't use a project ID.</p>
<code>--project-ID</code>	<p>The ID for an existing project.</p> <p>This option is required if you don't use a project name.</p>
<code>-s, --severity <SeverityLevel></code>	<p>A Contrast vulnerability severity level that returns a build failure status code that you can use to gate builds in pipelines.</p> <p>Valid values are: <code>critical</code>, <code>high</code>, <code>medium</code>, <code>low</code>, and <code>note</code>.</p> <p>The specified value is the minimum level of severity that returns a build failure status code. For example, if you specify <code>--severity high</code>, a finding of that severity or higher returns a build failure status code.</p> <p>This option applies to new and open vulnerabilities.</p>
<code>-q, --code-quality</code>	<p>Includes code quality rules when scanning source code.</p> <p>This option is not applicable for Java binary scans.</p>
<code>-r <ResourceGroupName></code>	<p>The name of the resource group where you want to add the project.</p> <p>If you are a hosted customer and role-based access control (page 1277) is turned on, this option is required.</p>
<code>--timeout <minutes></code>	<p>Lets you control the maximum time the source code scan engine scans the provided code. Specify a value in minutes.</p> <p>This option is only available for the Scan local engine.</p> <p>This option applies to each language found in the code to be scanned. For example if you set this to 120 minutes and your repo contains four languages, there is a potential for the scan to run for hours (4 languages x 120 minutes each).</p>

Exit codes

The Scan local engine returns these exit codes when a scan completes:

Exit code	Description
0	Scan completed successfully and uploaded results to Contrast

Exit code	Description
1	Input validation error
2	Error connecting to Contrast API server
3	Contrast API error returned
4	Scan local engine returned an error, details are in the log files
5	Unexpected error occurred, details are in the log files

Exclude files and folders for Contrast Scan

You have the option to exclude specified files or folders from scans. This feature is useful when you want to exclude artifacts that generate a lot of noise or are irrelevant to the scan.

Contrast [excludes some files and folders by default \(page 883\)](#).

Before you begin

- This feature is available only for multi-language source code scans with the Scan local engine.
- If you exclude files from a previously scanned project, Contrast changes the status of vulnerabilities affected by the exclusions to **Remediated**. For example, after you exclude files, the number of vulnerabilities in the scan results could be reduced from the original findings and the number of remediated vulnerabilities could increase.
- Specified file and folder names are case-sensitive.

Steps

1. In the root folder of the source code you are scanning, create a file named `.contrast-scan.json`.
2. In the JSON file, specify the files and folders you want to exclude using this format:

```
{
  "excludes": [
    "**/MavenWrapperDownloader.java",
    "**/*.js"
  ]
}
```

Replace the examples of `MavenWrapperDownloader.java` and `*.js` with the names of your files and folders.

Pattern examples

These examples show how you can specify excluded files and folders.

Patterns are considered relative paths.

This pattern example:	Excludes...
<code>*.java</code>	Files with zero or more than one character before the <code>.java</code> extension. For example: <code>.java</code> , <code>x.java</code> , and <code>FooBar.java</code> . Not excluded: Files that are not in the root directory being scanned, even if the filename has an extension of <code>.java</code> .
<code>? .java</code>	Files with one character before the <code>.java</code> extension For example: <code>x.java</code> or <code>A.java</code> . Not excluded: Files such as <code>.java</code> or <code>xyz.java</code> because they have zero or more than one character before the <code>.java</code> extension.
<code>**/*.java</code>	All folders and files with the extension of <code>.java</code> .
<code>**/CVS/*</code>	All files in <code>CVS</code> directories that exist anywhere in the directory tree.

This pattern example:	Excludes...
org/apache/ jakarta/**	All files in the org/apache/jakarta directory tree. Not excluded: The file org/apache/xyz.java because jakarta is not included in the path
org/ apache/**/CVS/ *	All files in CVS directories that are located anywhere in the directory tree under org/apache. Not excluded: A file named org/apache/CVS/foo/bar/Entries because foo/bar/ does not match the pattern.
/test/	All files that have test in their path, including test as part of a filename.
/*test/**	Excludes the string test wherever it is found in the path.

Default exclusions

By default, Contrast Scan excludes these files, folders, patterns, and extensions:

Excluded file and folder patterns	Excluded extensions	Excluded files
<ul style="list-style-type: none"> • /src/test/ • /__MACOSX/ • /*.min.js, • /.Designer.vb • **/.designer.vb • /*Reference.vb • /Service.vb • /*Silverlight.vb • /.Designer.cs • /*.designer.cs • /Reference.cs • /*Service.cs • /Silverlight.cs • **/. • /Pods/BuildHeaders/**/*.h • /Pods/Headers/**/*.h • /node_modules/ • /bower_components/ • /target/ • /bin/ • /obj/ • /dist/ • /lib/ 	<ul style="list-style-type: none"> • exe • dll • so • bin • arc • arj • zip • rar • ear • tar • tgz • gz • gzip • z 	<ul style="list-style-type: none"> • readme • changelog • changes • todo • license • copying • maintainers • thumbs.db

See also

[Directory-based tasks](#) provides additional information on patterns for specifying files and folders.

Create custom Scan rule exclusions

Create custom Scan rule exclusions when you are confident that a specific rule is generating multiple false positives and you are frequently setting the status for these findings to **Not a Problem**.

Before you begin

- Before you exclude rules, run a baseline scan to determine which rules are creating false positives
- Excluding rules affects the findings for the scan project.

Location of Scan rules

The [Contrast Scan rules \(page 884\)](#) section links to tables where you can find the rules for each language. Alternatively, during a scan, you can find the rules in target/engines/sast-engine4/rulesets under the source directory that you're scanning with the Scan local engine.



IMPORTANT

The `sast-engines4` folder and its subfolders are available temporarily when a scan is in progress.

For easy access, make a copy of the `rulesets` folder. This folder contains security rules files for each language (`gaking_{lang}_security.xml`) that the Scan local engine supports. To find the name of a rule in a file, search for `<rule name " "`.

Steps

1. Create a text file called `contrastsec.checks.config` and place it at the root of the project you are going to scan.

The format of the file is:

```
[rule-Engine-rule-ID]
active=false
```

2. To exclude multiple rules, repeat the block of lines with an empty row between each block. For example, to exclude the **Detect and handle input/output errors** and **Don't use cast** rules for CPP, the file looks like this:

```
[OPT.CPP.CERTC.FIO33]
active=false

[OPT.CPP.DontUseCast]
active=false
```

Effects of Scan rule exclusions

When you disable rules using the `contrastsec.checks.config` file, the status for the findings corresponding to the excluded rule change to **Remediated**.

If you reenable the rule using the `contrastsec.checks.config` file or remove the file, the status for the findings corresponding to the newly enabled rule changes to **Reopened**.

Contrast Scan rules

These tables include the supported Contrast Scan rules: test

ABAP (page 675)	Go (page 745)	NATURAL (page 807)	Scala (page 844)
ActionScript (page 687)	HTML (page 748)	Objective-C (page 808)	SQL (page 846)
ASP (page 691)	Informix (page 751)	Oracle Forms (page 816)	SQLScript (page 849)
ASP.NET (page 689)	Java (page 753)	PHP (page 818)	Swift (page 850)
C (page 734)	JavaScript (page 789)	PL/SQL (page 827)	Transact-SQL (page 854)
C# (page 692)	JCL (page 800)	PowerScript (page 832)	VB.NET (page 857)
COBOL (page 707)	JSP (page 801)	Python (page 834)	Visual Basic 6 (page 871)
CPP (page 721)	Kotlin (page 803)	RPG4 (page 840)	XML (page 873)

Use Contrast Scan for GitHub repositories

Use the [Contrast Local Scan](#) to scan GitHub repositories for vulnerabilities without uploading your files to Contrast.

Before you begin

- Get your Contrast authentication details by selecting the **user menu > User settings** in the Contrast web interface. You need these details:
 - Organization ID
 - API key
 - Service key
- You also need a valid Contrast username and the URL of your Contrast instance.

Steps

1. Configure these GitHub secrets:
 - CONTRAST__API__API_KEY
 - CONTRAST__API__ORGANIZATION
 - CONTRAST__API__SERVICE_KEY
 - CONTRAST__API__USER_NAME
 - CONTRAST__API__URL
2. Create a workflow or update an existing one to run this action against your code. This example shows how to run the action on push.

```
name: Scan with local scanner

on:
  push:
    branches:
      - 'main'

permissions:
  contents: read

jobs:
  scan:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - uses: Contrast-Security-OSS/contrast-local-scan-action@v1.0.0
        with:
          apiUrl: ${ secrets.CONTRAST__API__URL }
          apiUserName: ${ secrets.CONTRAST__API__USER_NAME }
          apiKey: ${ secrets.CONTRAST__API__API_KEY }
          apiServiceKey: ${ secrets.CONTRAST__API__SERVICE_KEY }
          apiOrgId: ${ secrets.CONTRAST__API__ORGANIZATION }
```

The [README](#) contains additional examples of other inputs that you can add to your workflow.

Required inputs

- `apiUserName`: A valid Contrast username.
- `apiKey`: Your Contrast API key.
- `apiServiceKey`: The Contrast service key.
- `apiOrgId`: The Contrast organization ID,

Optional inputs

- `apiUrl`: The URL of your Contrast instance.

The default value is: `https://app-agents.contrastsecurity.com/Contrast`.

- `checks`: If set, adds GitHub checks to the current commit based on any vulnerabilities found.
- `codeQuality`: Passes the `-q` option to the Contrast local scan engine to include code quality rules in the scan.
- `label`: A label to associate with the current scan.
The default value is the current ref, for example: `refs/heads/main`.
- `memory`: Memory setting passed to the underlying scan engine. The default value is 2G.
- `path`: The path the Contrast local scan engine uses for the scan.
The default value is the current repository path.
- `projectName`: The name of a project to associate with the scan.
The default value is the current GitHub repository name, for example, `mycompany/myrepo`
- `resourceGroup`: Passes the `-r` option to the Contrast local scan engine to associate newly created projects with the specified resource group.
- `severity`: Set this to cause a build to fail if the scan finds vulnerabilities at this severity or higher.
Valid values are critical, high, medium, low, note.

See also

Contrast [Contrast Local Scan](#) for latest details.

[Contrast Scan local engine \(page 672\)](#)

Scan languages with the Semgrep engine

Contrast provides, as a courtesy to those customers with Terraform, Rust, and Ruby 3.X source code elements, an optional way to scan their source code using the Semgrep open source SAST scanner and have results presented in the Contrast web interface along with other supported languages.

Steps

1. Download the Semgrep engine from [Semgrep](#).
2. Place the Semgrep engine file in the same location as the Scan local engine JAR file.
3. Run a scan with the Contrast Scan local engine.
When the Contrast Scan local engine detects the presence of Terraform (TF) files, Rust (RS) files, or Ruby (RB) files, it passes the relevant files to the Semgrep engine. The Scan local engine uses the Semgrep rules for those languages ([Terraform rules](#), [Rust rules](#), or [Ruby rules](#)) and creates a SARIF file for the scanned languages.
When scanning of the complete repo completes:
 1. The scan process combines the SARIF files that the Semgrep engine and the Contrast Scan local engine create.
 2. The scan process uploads the SARIF file to the Contrast web interface.
4. View results as described in the [Analyze scan results \(page 887\)](#) section.

Contrast support

Contrast provides support for Rust, Terraform, and Ruby on an as is basis. You are free to use the Semgrep engine to scan Rust, Terraform, and Ruby files (and any other language) without integration with the Contrast SAST platform. Contrast supports this functionality for convenience purposes only, where Rust, Terraform, or Ruby files are part of a larger repo.

View local scan results

View details and results of a local scan in the Contrast web interface.

If you specified a location for output results, you can view the SARIF file that the scan created on your local system.

Steps

1. Log in to the Contrast web application.
2. Select the **Scans** tab.
3. In the scans list, select the scan project for the local scan.
4. To view results from the local scan, explore the Overview, Vulnerabilities, and Policy tabs.
The Contrast Documentation contains additional information on [analyzing scan results](#).

Analyze scan results

A scan observes the data flow in an application and reports vulnerabilities that it discovers.

After you analyze the results, update your code and run the scan again to verify the vulnerability is fixed.

Scoring

Contrast provides a score for each scan that represents the potential security risk for an application:

- Scoring is not dynamic except for vulnerabilities that have a status of **Not a Problem**.
[Configure dynamic scoring \(page 664\)](#) describes how to use dynamic scoring. For all other vulnerabilities, run the scan again to see updated scores after updating code.
- Scoring uses the standard scoring methodology described in the [Contrast application scoring guide \(page 1269\)](#).

Before you begin

[Contrast Scan vulnerability workflow \(page 893\)](#) provides guidelines for managing vulnerabilities that Contrast Scan reports.

Steps

After a scan completes, you can view information on vulnerabilities, the name of the project creator, and the name of the person who ran each scan.

1. Select **Scans** in the header.
The Scans page shows a list of scan projects..
2. Select a scan project.
3. On the Overview tab, view a summary of the scan results as well as a list of scans in the project.

MyTest45
Language: Java | Last Scan: 6 Minutes Ago | Created By: Jane Danielson

Overview Vulnerabilities Policy + New Scan

F 0/100
34 Vulnerabilities 0 New Vulnerabilities 0 Remediated 4 Scans Completed 0 Days Since Last Scan

SCAN HISTORY

All (4)

VULNERABILITIES	LABEL	SCAN DATE	NAME	LANGUAGE	COVERAGE
(37) 7	2023-10-31 14:56:22	2 minutes ago	Jane Danielson	Java	View
(37) 7	2023-10-31 14:51:50	7 minutes ago	Jane Danielson	Java	View
(37) 7	2023-10-31 14:48:50	10 minutes ago	Jane Danielson	Java	View
(37) 7	2023-10-31 14:43:04	16 minutes ago	Jane Danielson	Java	View

< Page 1 of 1 > Results per page 10 25 50

The summary section shows these details:

- **Score:** A letter grade that represents the potential security risk for application based on the most recent scan in the project.
- **Vulnerabilities:** The number of vulnerabilities discovered in the most recent scan. To see details about discovered vulnerabilities, select the number.
- **New Vulnerabilities:** The number of new vulnerabilities discovered in the most recent scan. This value excludes vulnerabilities that previous scans discovered and are not fixed.

For example:

If Scan 1 discovered three vulnerabilities:

- The number of vulnerabilities is three.
- The number of new vulnerabilities is also three.

If code modifications introduce a new vulnerability but do not fix existing vulnerabilities, when you run Scan 2:

- The vulnerabilities number changes to four (all discovered vulnerabilities).
- The new vulnerabilities number becomes one (the new one discovered in Scan 2).

To see details about new vulnerabilities, select the number.

- **Remediated:** The number of vulnerabilities that are fixed by changing source code or configuration files within the application.
To see details about remediated vulnerabilities, select the number.
- **Scans completed:** The number of scans completed in the project.
To see details about completed scans, select the number.
- **Days since last scan:** The number of days since the last scan completed.

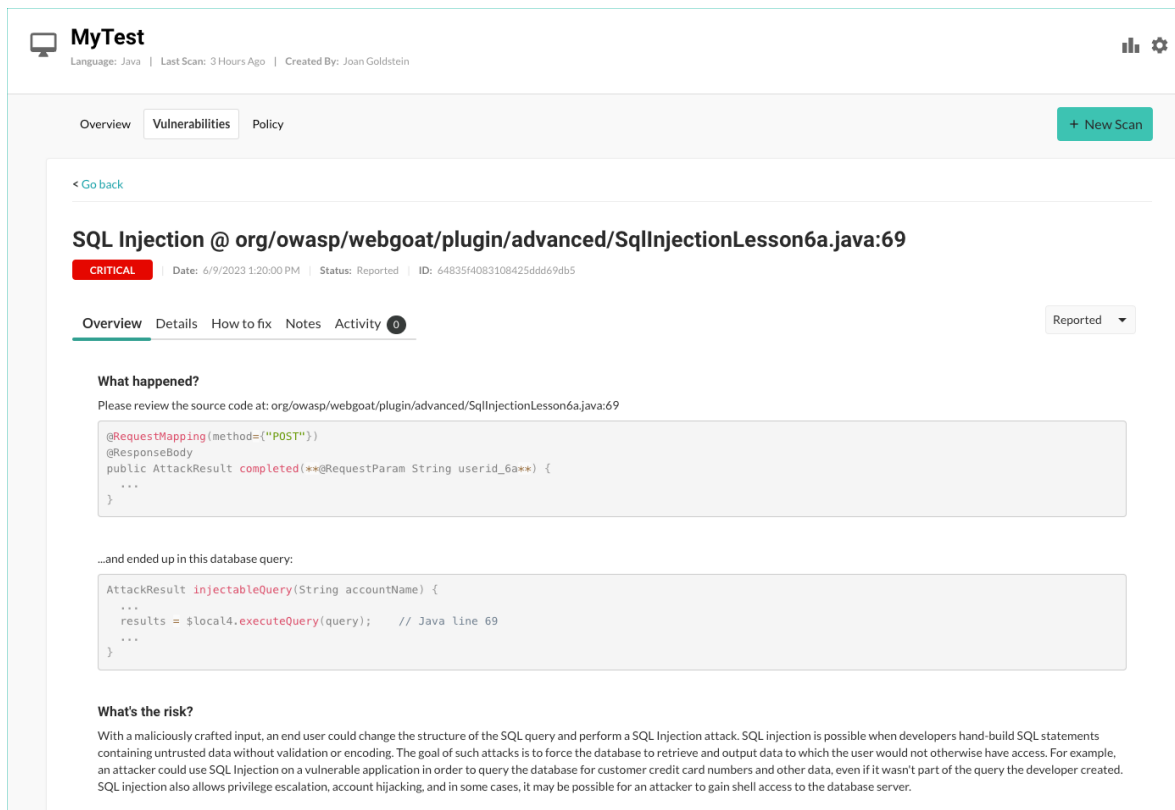
The Scan history shows these details:

- **Vulnerabilities:** A bar that shows the different types of detected vulnerabilities for a scan.
To view a filtered list of a specific type of vulnerability, select a section of the bar.

- **Label:** The label associated with the scan.
To view additional [scan details \(page 890\)](#), select the label.
- **Scan date** The date the scan completed.
- **Name:** The name of the person who ran the scan.
- **Language:** The languages detected in the scanned code.
- **Coverage:** A link to additional [scan details \(page 890\)](#).

To export the results to a SARIF file, select the **Download** icon () at the end of a scan's row.

- To view more information about a specific vulnerability, select the **Vulnerabilities** tab and select the vulnerability.



The screenshot shows the Contrast web interface for a scan named "MyTest". The interface has tabs for Overview, Vulnerabilities, and Policy. The Vulnerabilities tab is active, showing a list of vulnerabilities. One vulnerability is selected: "SQL Injection @ org/owasp/webgoat/plugin/advanced/SqlInjectionLesson6a.java:69". The vulnerability is marked as "CRITICAL". Below the title, there are tabs for Overview, Details, How to fix, Notes, and Activity. The Overview tab is active, showing a description of the vulnerability, the source code snippet where it was found, and the risk associated with it.

SQL Injection @ org/owasp/webgoat/plugin/advanced/SqlInjectionLesson6a.java:69

CRITICAL | Date: 6/9/2023 1:20:00 PM | Status: Reported | ID: 64835f4083108425dd69db5

Overview Details How to fix Notes Activity 0

What happened?

Please review the source code at: [org/owasp/webgoat/plugin/advanced/SqlInjectionLesson6a.java:69](#)

```
@RequestMapping(method={"POST"})
@ResponseBody
public AttackResult completed(**@RequestParam String userid_6a**) {
    ...
}
```

...and ended up in this database query:

```
AttackResult injectableQuery(String accountName) {
    ...
    results = $local4.executeQuery(query); // Java line 69
    ...
}
```

What's the risk?

With a maliciously crafted input, an end user could change the structure of the SQL query and perform a SQL Injection attack. SQL injection is possible when developers hand-build SQL statements containing untrusted data without validation or encoding. The goal of such attacks is to force the database to retrieve and output data to which the user would not otherwise have access. For example, an attacker could use SQL Injection on a vulnerable application in order to query the database for customer credit card numbers and other data, even if it wasn't part of the query the developer created. SQL injection also allows privilege escalation, account hijacking, and in some cases, it may be possible for an attacker to gain shell access to the database server.

- The Overview tab for the selected vulnerability shows a description of the vulnerability, including what happened in your code and the risk associated with the vulnerability.
- To view the details about the vulnerability and its location in your code, select the **Details** tab:
 - The method where a vulnerability exists.
 - The file where the scan discovered the vulnerability.
 - The first line in the code where the scan discovered the vulnerability.
 - To view suggestions for fixing the code, select the **How to fix** tab.
 - To view suggestions for fixing the code from Secure Code Warrior, select the **SCW How to fix** tab.
This tab includes suggestions from Secure Code Warrior as well as videos, links to additional information, and guidelines. If no relevant Secure Code Warrior information is available, this tab is not displayed.
 - To view additional details about the vulnerability, select the **Notes** tab for these details:
 - When the vulnerability is detected
 - The code module where Contrast found the vulnerability
 - The type of vulnerability (for example, injection)
 - Severity
 - Risk confidence

- Security standards that apply to the vulnerability
- e. To view vulnerability activity, select the **Activity** tab for these details:
- The user who made changes
 - Vulnerability status changes
 - Comments

See also

[View scan details \(page 890\)](#)

View scan details

Scan details include:

- A summary of the scan results.
- Scan coverage details.

Before you begin

- Identify the scan project that contains the scan you want to view.

Steps

1. In the header, select **Scans**.
2. Select a scan project.
3. At the top of the page, view these details:
 - The language of the scanned artifact.
Composite indicates that the scan discovered multiple languages in the artifact.
 - The last time a scan occurred for this project.
 - The name of the person who ran the scan.
 - Metadata associated with the project, if configured.
 - If role-based access control is turned in, the name of the resource group with which the project is associated, based on the role of the logged-in user.
Different users could see different resource groups, depending on their role. For example, a standard user could see different resource groups than a user with administrator permissions.
4. View the summary details:
 - **Score**: A letter grade that represents an aggregate of both the number and seriousness of the vulnerabilities discovered based on the amount of the application that was exercised.
 - **Vulnerabilities**: The current number of vulnerabilities.
 - **New Vulnerabilities**: The number of new vulnerabilities discovered with the most recent scan.
 - **Remediated**: The number of vulnerabilities that were remediated:
 - **Scans completed**: The number of scans in the project.
 - **Days since last scan**: The number of days since the last scan completed.
5. To view additional details about vulnerabilities, select the **Vulnerabilities** tab at the top of the page.

MyTest Language: Java Last Scan: 1 Minute Ago Created By: Joan Goldstein						
Overview Vulnerabilities Policy						+ New Scan
<input type="checkbox"/>	SEVERITY ▼	VULNERABILITY ▼	LANGUAGE ▼	CWE	LAST DETECTED	STATUS ▼ Clear
<input type="checkbox"/>	CRITICAL ▼	SQL Injection @ org/owasp/webgoat/plugin/advanced/SqlInjectionLesson6a.java:69	Java	89	1 minute ago	Reported
<input type="checkbox"/>	CRITICAL ▼	SQL Injection @ org/owasp/webgoat/plugin/mitigation/Servers.java:46	Java	89	1 minute ago	Reported
<input type="checkbox"/>	CRITICAL ▼	SQL Injection @ org/owasp/webgoat/plugin/challenge5/challenge6/Assignment5.java:49	Java	89	1 minute ago	Reported

- To filter the view by severity, vulnerability, language, CWE, or status, select the **Filter** icon (▼) next to the column heading and select available options.
 - In addition to filtering by severity, you can also [edit the severity \(page 896\)](#).
 - The Vulnerability column shows the name of the vulnerability and where in the code it was found.
 - The Language column shows the language of the code where the scan found the vulnerability.
 - The CWE column shows the number of the CWE that maps to the rule for the vulnerability. For example, if CWE-89 maps to a rule for a specific vulnerability, the CWE column displays 89. If no CWE exists for a specific vulnerability, the column is blank.
 - The Status column shows the status of the vulnerability.
[Scan statuses \(page 893\)](#) provides descriptions of each vulnerability status.
 - To [download the vulnerability data to a CSV file \(page 897\)](#), select the **Download** icon (⬇) at the top of the vulnerability list.
6. To view the rules that the scan used, select the **Policy** tab at the top of the page.
 7. To view coverage details, under Scan history, select the link in the Label column for a specific scan or select **View** in the Coverage column under Scan history.

The screenshot shows the Contrast Scan interface for a scan named "webgoat-server-8.0.0.M21.jar". At the top, there's a header with "MyTest" and a settings icon. Below the header, there are tabs for "Overview", "Vulnerabilities", and "Policy". A "New Scan" button is in the top right. The main content area shows scan details: "Scan Date: 6/9/2023 1:19:09 PM", "Created By: Jane Doe", and "ID: 45be7ecd-44d1-4ff5-a04a-aa4b7a76022a". There's a "Scan Status" section showing "Completed in 1 minute" and "Policy: Default". A "Custom Code" section shows "Automatically Detected". An "Analysis" section displays four metrics: "111 Entry Points", "296 Dangerous Calls", "512 Custom Classes", and "3829 Library Classes". To the right is a donut chart titled "Breakdown" showing "Vulnerabilities" (red), "Warnings" (teal), and "Other findings" (grey). Below this are tabs for "Vulnerabilities" and "Other Findings". A table lists vulnerabilities with columns for "Severity", "Vulnerability", and "Status". Two vulnerabilities are shown, both with a "CRITICAL" severity and "Reported" status.

Severity	Vulnerability	Status
CRITICAL	SQL Injection @ org/owasp/webgoat/plugin/introduction/SqlInjectionLesson5b.java:69	Reported
CRITICAL	SQL Injection @ org/owasp/webgoat/plugin/advanced/SqlInjectionChallenge.java:51	Reported

- At the top of the list, view a summary of scan details.
The name of the person who ran the scan is displayed at the top of the summary.

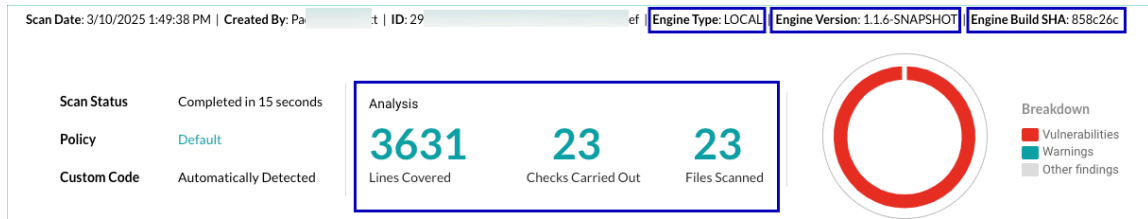


NOTE

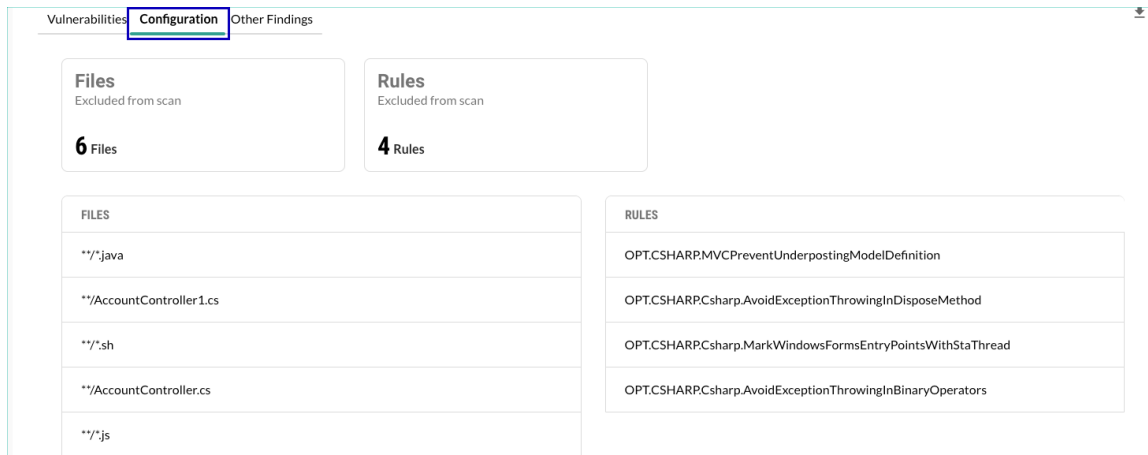
The ability to view the name of the person who ran the scan is available for scan projects created after June 12, 2023.

Scans with the Contrast Scan local engine: If you used the Contrast Scan local engine, the summary shows the type of scan engine (for example LOCAL), the version of the scan engine used, and the engine SHA checksum.

The charts in the Analysis section show the number of lines of code scanned, the number of rules that the scan checked, and the number of files scanned.



- Under **Vulnerabilities**, view the severity, vulnerability name, and status of the vulnerability. Contrast has high confidence that the vulnerabilities in this list require remediation.
- Under **Configuration**, view the files and rules excluded from the scan. To see this tab, use the Contrast Scan local engine and [exclude files \(page 882\)](#) or [rules \(page 674\)](#).



- Under **Other Findings**, view additional vulnerabilities that the scan found. Due to the assumptions the scan made when reporting these vulnerabilities, Contrast has lower confidence that the vulnerabilities in this list require remediation

View details for scanned branches

The Scan local engine lets you scan personal branches in repos. You can view the results of these scans in the Contrast web interface.

Before you begin

- Scan results for branches have no affect on the score or aggregated results for the parent scan project.

Steps

1. With the [Scan local engine \(page 877\)](#), scan a branch.
2. In the Contrast web interface, select **Scans**.
3. Select the scan project associated with your branch.
4. Select the **Branches** tab.
This tab is available only if the scan project has scanned branches associated with it. Use the Scan local engine to [scan branches \(page 880\)](#).
5. View the details of the scan:
 - **Overview**: Shows a summary of the scan results and the scan history.
 - **Vulnerabilities**: Shows the list of vulnerabilities for the scanned branch. Select a vulnerability to view additional details.
 - **Policy**: Shows the list of rules applied to the scanned branch.

- To return to the parent scan project, select **Go to parent project** at the top of the page.

Contrast Scan vulnerability statuses

This table lists the different statuses that you or Contrast sets for vulnerabilities that Contrast Scan discovers.

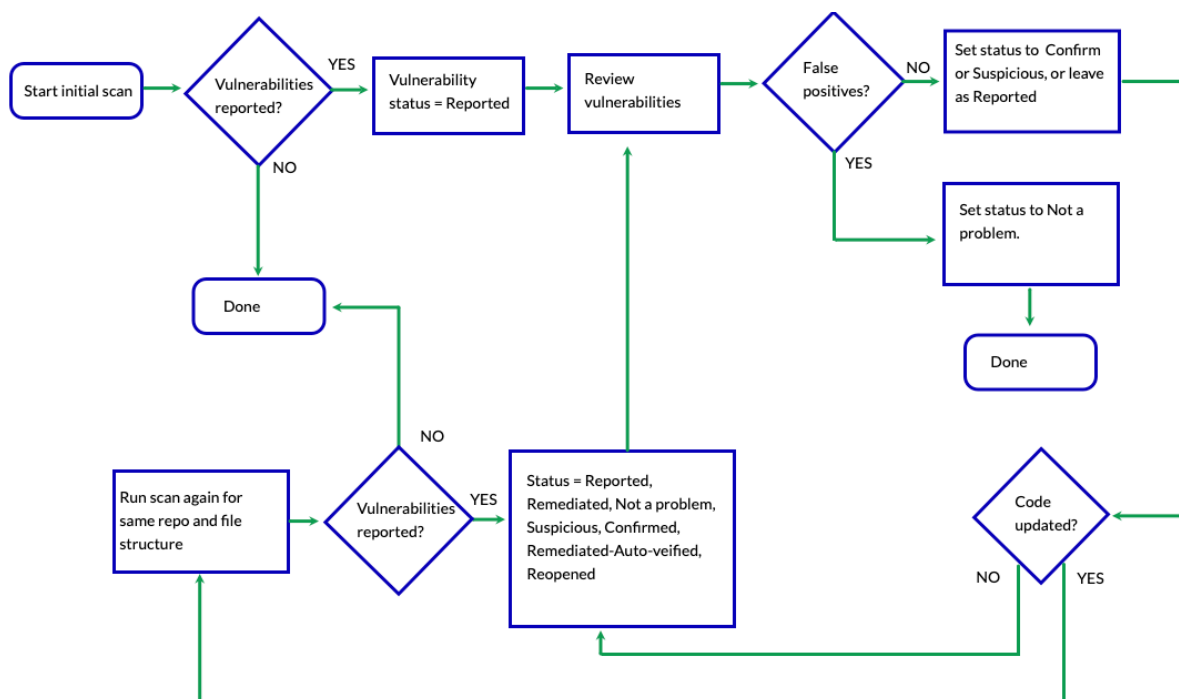
Status	Automated or manually set?	Description
Reported	Automated	Contrast sets this status automatically the first time it discovers vulnerabilities during a scan.
Confirmed	Manual	You've reviewed the code and confirmed that the vulnerability is a true finding.
Suspicious	Manual	The vulnerability seems to be a true finding, but it needs more investigation to determine its validity.
Not a problem	Manual	The vulnerability doesn't require code changes. Optionally, you can provide a reason for this status change. If you change the status to Not a Problem , it never changes to Remediated or any other status, even if subsequent scans don't discover the vulnerability. To have the vulnerability assessed again, change the status to Confirmed or Suspicious .
Remediated	Automated	A change to the source code or application configuration files fixed the vulnerability.
Fixed	Automated	Not currently used.
Remediated Auto-verified	Automated	The vulnerability had a status of Remediated . After the fifth scan, Contrast updates the Remediated status to Remediated Auto-Verified .
Reopened	Automated	The vulnerability had a status of Remediated , but a new scan detects the vulnerability again. Contrast changes the status to Reopened .

Contrast Scan vulnerability workflow

This topic provides workflow guidelines for managing the vulnerabilities that Contrast Scan reports.

Diagram: Scan vulnerability workflow

This diagram shows a suggested workflow for managing vulnerabilities that Contrast Scan reports.



Workflow for managing scan vulnerabilities

1. Create a scan project and run the scan.
Each time you run a scan for a specific project, ensure you are using the same repo and file structure.
Contrast sets the status for new vulnerabilities to **Reported**.
2. Review the findings.
 - If the reported vulnerabilities are false positives or do not require code changes, change the status to **Not a Problem** and add a comment.
 - If the finding is a true positive, change the vulnerability status to **Confirmed** to indicate that the development team should fix the code.
 - If the finding is true based on the details of the vulnerability, but the vulnerability requires additional investigation to determine its validity, changing the status to **Suspicious** might be appropriate.
3. Fix the code or exclude irrelevant files and folders.
4. Run the scan again.
 - If the vulnerability is not found in a subsequent scan, Contrast changes its status to **Remediated**.
 - If Contrast detects a vulnerability with a status of **Not a Problem**, **Confirmed**, or **Suspicious** in a subsequent scan, Contrast keeps the current status.
 - If you exclude files or folder file related to a previously reported vulnerability and the vulnerability had a status of **Confirmed** or **Suspicious**, Contrast changes the status to **Remediated** if the scan no longer detects the vulnerability.
 - If a vulnerability had a status of **Remediated** and the vulnerability is detected in a subsequent scan, Contrast changes the status to **Reopened**.
 - If the vulnerability has a status of **Remediated** for five scans, Contrast changes the status to **Remediated Auto-Verified**.
5. Repeat steps 2 through 4 until you address all the vulnerabilities.

Edit scan vulnerability status

When Contrast discovers a vulnerability during a scan, it assigns a status of **Reported** to the vulnerability. This status indicates that the vulnerability could possibly be exploited.

You can change this status, based on how you are managing the vulnerability, to one of these values:

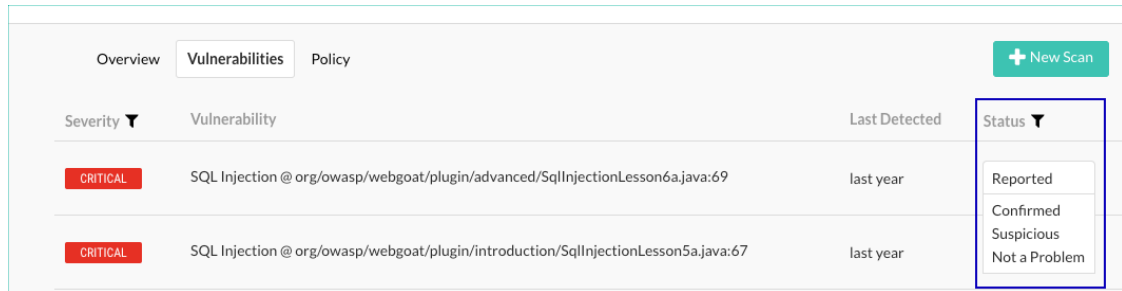
- **Confirmed:** You've confirmed that the vulnerability is a true finding by reviewing the source code or exploiting it.
- **Suspicious:** You've confirmed that the vulnerability appears to be a true finding based on the details provided, but it requires more investigation to determine its validity.
- **Not a problem:** You've determined that the vulnerability doesn't require code changes.
If you change the status to **Not a Problem**, it never changes to **Remediated** or any other status, even if subsequent scans don't discover the vulnerability. To have the vulnerability assessed again, change the status to **Confirmed** or **Suspicious**.

[Batch edit Scan vulnerability status \(page 895\)](#) describes how to edit multiple statuses at the same time

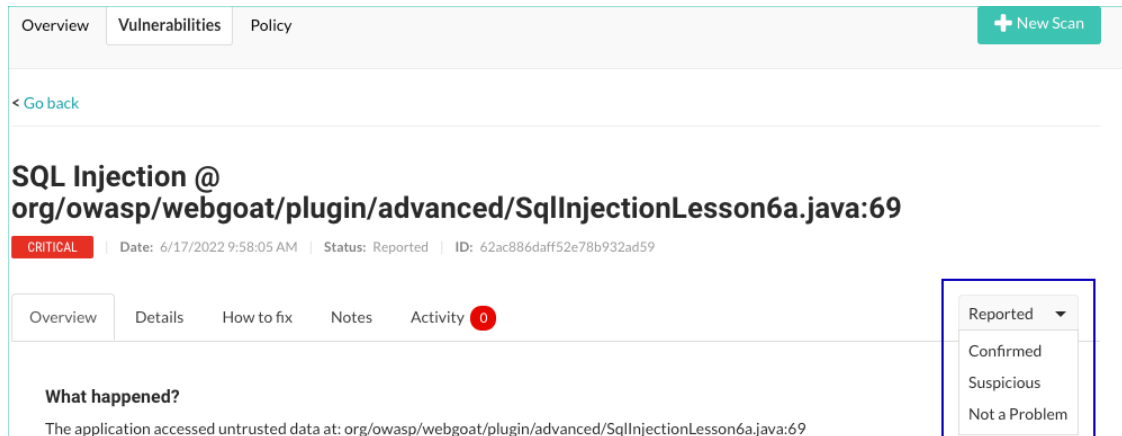
Steps

1. Select **Scans** in the header.
2. Select a Scan project.
3. Select the **Vulnerabilities** tab.
4. Change the status:

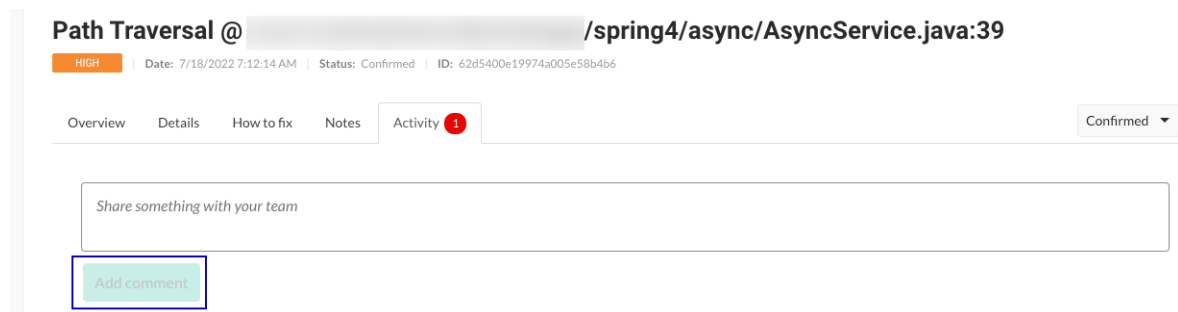
- a. On the Vulnerabilities page, select a status in the Status column.



Alternatively, select a vulnerability from the Vulnerabilities list and select a status on the right side of the view.



- b. Optionally, enter a comment explaining why you are making the change and select **Override**.
5. Add comments for a vulnerability without changing its status:



- a. From the Vulnerabilities tab, select a vulnerability.
- b. Select the **Activity** tab.
- c. Enter a comment and select **Add comment**.

Batch edit Scan vulnerability status

When Contrast discovers a vulnerability during a scan, it assigns a status of **Reported** to the vulnerability. This status indicates that the vulnerability could possibly be exploited.

You can change this status for multiple vulnerabilities, based on how you are managing them, to one of these values:

- **Confirmed:** You've confirmed that the vulnerability is a true finding by reviewing the source code or exploiting it.
- **Suspicious:** You've confirmed that the vulnerability appears to be a true finding based on the details provided, but it requires more investigation to determine its validity.
- **Not a problem:** You've determined that the vulnerability doesn't require code changes.

If you change the status to **Not a Problem**, it never changes to **Remediated** or any other status, even if subsequent scans don't discover the vulnerability. To have the vulnerability assessed again, change the status to **Confirmed** or **Suspicious**.

Steps

1. Select **Scans** in the header.
2. Select a Scan project.
3. Select the **Vulnerabilities** tab.
4. Use the checkbox on the left to select multiple vulnerabilities with the same status. You can select vulnerabilities with different types.
5. In the batch action menu at the bottom of the page, select **Status** and select a status from the dropdown.
6. If you selected one or more vulnerabilities of the same type, optionally select the checkbox to change the status of all vulnerabilities that match that type.

Mark as Confirmed

8 Selected Include all 8 Command Injection types? ☒ Select all

Development confirmed that these are real issues

Cancel Change status

7. Optionally, enter a comment in the Mark window.
8. Select **Change status**.



NOTE

Changing the status for large numbers of vulnerabilities at the same time can take several minutes to complete. You can continue to make changes without waiting. Contrast displays a message when the change operations finish.

Edit scan vulnerability severity

Contrast classifies scan vulnerabilities based on the likelihood and impact of a vulnerability in the code, from most to least severe:

- Critical
- High
- Medium
- Low
- Notes

You have the option to change the severity that Contrast applies automatically to a vulnerability. Future scans do not override a severity change.

Before you begin

- If you are using role-based access control, a role with the View, edit, and delete project action is required.
- If you are using organization users and groups, the Organization Admin role is required.

Steps

1. In the header, select **Scans**.
2. Select a scan project.
3. Select the **Vulnerabilities** tab.
4. Select the colored badge in the Severity column and choose a new level from the menu.
 - a. If more than one vulnerability of the same type exists, you have the option of changing the severity for the selected vulnerability only or the severity of all matching vulnerabilities.
 - b. Optionally, add a reason for the change in the Comment box.
 - c. Select **Change severity**.

Download scan results

You can download scan results to these file types:


- **SARIF file:** After a scan completes, you can download the results to a Static Analysis Results Interchange Format (SARIF) file. This type of file is a standard, JSON-based format for the output of static analysis data.

To optimize storage usage, SARIF files are available for up to five days after the scan completes. No download option is available for older scans.
- **CSV file:** You can download vulnerability data for a scan project to a Comma-separated Values (CSV) file. The CSV file:
 - Excludes vulnerabilities with a status of **Remediated** and **Not a problem**.
 - Includes or excludes data based on selected filters.

Before you begin

Identify the scan whose results you want to download.

Steps for SARIF download

1. Select **Scans** in the header.
2. Select a scan project.
3. In the Overview tab, under Scan history, select the **Download** icon () at the end of a scan row.
4. When prompted, select a location for the SARIF file.

Steps for CSV download

1. Select **Scans** in the header.
2. Select a scan project.

3. Select the **Vulnerabilities** tab.
4. To filter the data in the CSV file, select any of the filters for **Severity**, **Vulnerability**, **Language**, or **Status**.
5. Select the **Download** icon (↓) at the top of the page.
6. If the number of vulnerabilities exceeds 2,000, you are prompted to select a page of results to download.
You can download one page at a time.
7. When prompted, select a location for the CSV file.

SARIF file data

When you download results, Scan writes the data to a SARIF file.

SARIF is a standard data model and serialization format for static analysis results. Understanding the data in the SARIF file can help when you need a deeper understanding of scan results.

The SARIF file includes this type of information:

- Information about the scanner that Contrast uses
- Data on what was scanned and the scan composition
- Data on vulnerability findings
- Errors or notifications that are handled gracefully during the scan
- Scan coverage data

Scanner data

This example shows data about the scanner that Contrast uses:

```
5  "tool" : {  
6    "driver" : {  
7      "name" : "Contrast Scan",  
8      "organization" : "Contrast Security, Inc.",  
9      "version" : "pkg: 2.0.0-SNAPSHOT, engine: 2.0.0-SNAPSHOT, policy: 2.0.0-SNAPSHOT",  
10     "informationUri" : "https://www.contrastsecurity.com"  
11   }  
}
```

Vulnerability data

The examples in this section show some of the data for a single vulnerability. The scan shows this data in the `results` section.

• Single object

This example shows the scan results for a SQL injection vulnerability. The data in the `threadFlows` section show the scanned data from the source to the data sink.

```

13  "artifacts" : [ ],
14  "results" : [ {
15    "ruleId" : "sql-injection",
16    "level" : "error",
17    "message" : {
18      "text" : "sql-injection in User.fetch() reachable from LoginController.login()"
19    },
20    "locations" : [ {
21      "physicalLocation" : {
22        "artifactLocation" : {
23          "uri" : "file:///github/workspace/src/main/java/com/scalesec/vulnado/User.java"
24        },
25        "region" : {
26          "startLine" : 49,
27          "startColumn" : 1,
28          "snippet" : {
29            "text" : "public static User fetch(String un) {\n ... \n  rs = stmt.executeQuery(query);    // Java line
30              49\n ... \n}"
31          }
32        },
33        "contextRegion" : {
34          "snippet" : { }
35        }
36      } ],
37      "partialFingerprints" : {
38        "GITHUB_SOURCECODE_LSH/v1" :
39        "e86a7a0c38715f4a:1-158823731d54d12e:1-6b692237789f9a2a:1-64ed8e6526b79cf4:1-13db8c734146ff90:1"
40      },
41      "codeFlows" : [ {
42        "message" : {
43          "text" : "Untrusted data flow from LoginController.java:19 to User.java:49 via variable `query`"
44        },
45        "threadFlows" : [ {
46          "locations" : [ {
47            "location" : {
48              "physicalLocation" : {
49                "artifactLocation" : {
50                  "uri" : "com/scalesec/vulnado/LoginController.java"
51                },
52                "region" : {
53                  "startLine" : 19,
54                  "snippet" : {
55                    "rendered" : {
56                      "text" : "@CrossOrigin(origins={\"*\"})\n@RequestMapping(value=\"/login\"), method={\"POST\"},
57                        produces={\"application/json\"}, consumes={\"application/json\"})\nLoginResponse login(**@RequestBody
58                          LoginRequest input**) {\n ... \n}"
59                    }
60                  },
61                  "properties" : {
62                    "ir" : [ "@CrossOrigin(origins={\"*\"})\n@RequestMapping(value=\"/login\"), method={\"POST\"},
63                      produces={\"application/json\"}, consumes={\"application/json\"})\nLoginResponse login(**@RequestBody
64                        LoginRequest input**) {\", \" ...\", \"}\" ]
65                  }
66                }
67              } ],
68              "contextRegion" : {
69                "snippet" : { }
70              }
71            }
72          }
73        }
74      ]
75    }
76  ]
77 }

```

- **Sink location**

This example shows the sink or problem location for the vulnerability.

```

"locations" : [ {
  "physicalLocation" : {
    "artifactLocation" : {
      "uri" : "file:///github/workspace/src/main/java/com/scalesec/vulnado/User.java"
    },
    "region" : {
      "startLine" : 49,
      "startColumn" : 1,
      "snippet" : {
        "text" : "public static User fetch(String un) {\n ... \n  rs = stmt.executeQuery(query);    // Java line
          49\n ... \n}"
      }
    },
    "contextRegion" : {
      "snippet" : { }
    }
  }
]
}

```

- **Thread flow steps**

This example shows some of the data for one execution step in the data flow:

```

44     "threadFlows" : [ {
45         "locations" : [ {
46             "location" : {
47                 "physicalLocation" : {
48                     "artifactLocation" : {
49                         "uri" : "com/scalesec/vulnado/LoginController.java"
50                     },
51                     "region" : {
52                         "startLine" : 19,
53                         "snippet" : {
54                             "rendered" : {
55                                 "text" : "@CrossOrigin(origins={\"*\"})\n@RequestMapping(value=\"/login\"), method={\"POST\"},
produces={\"application/json\"}, consumes={\"application/json\"})\nLoginResponse login(**@RequestBody
LoginRequest input**) {\n    ...\n}"
56                             }
57                         },
58                         "properties" : {
59                             "ir" : [ "@CrossOrigin(origins={\"*\"})\n@RequestMapping(value=\"/login\"), method={\"POST\"},
produces={\"application/json\"}, consumes={\"application/json\"})\nLoginResponse login(**@RequestBody
LoginRequest input**) {\", \"    ...\", \"}\" ]
60                         }
61                     },
62                 },
63                 "logicalLocations" : [ {
64                     "name" : "LoginController.login()",
65                     "fullyQualifiedName" : "com.scalesec.vulnado.LoginController.login(com.scalesec.vulnado.LoginRequest)"
66                 } ]
67             }, {
68                 "location" : {
69                     "physicalLocation" : {
70                         "artifactLocation" : {
71                             "uri" : "com/scalesec/vulnado/LoginController.java"
72                         },
73                         "region" : {
74                             "startLine" : 20,
75                             "snippet" : {
76                                 "rendered" : {
77                                     "text" : "@CrossOrigin(origins={\"*\"})\n@RequestMapping(value=\"/login\"), method={\"POST\"},
produces={\"application/json\"}, consumes={\"application/json\"})\nLoginResponse login(@RequestBody
LoginRequest input) {\n    ...\n    $stack0 = input.username;    // Java line 20\n    user = User.fetch
($stack0);    // Java line 20\n    ...\n}"
78                                 }
79                             },
80                             "properties" : {
81                                 "ir" : [ "$stack0 = input.<com.scalesec.vulnado.LoginRequest:java.lang.String username>;    // Java
line 20\", \"user = staticinvoke <com.scalesec.vulnado.User: com.scalesec.vulnado.User fetch(java.lang.
String)>($stack0);    // Java line 20\" ]
82                             }
83                         }
84                     },
85                     "logicalLocations" : [ {
86                         "name" : "LoginController.login()",
87                         "fullyQualifiedName" : "com.scalesec.vulnado.LoginController.login(com.scalesec.vulnado.LoginRequest)"
88                     } ],
89                     "message" : {
90                         "text" : "un"
91                     }
92                 },
93             },
94             "state" : {
95                 "un" : {
96                     "text" : "tainted",
97                     "properties" : {
98                         "taintTags" : [ "untrusted", "cross-site" ]
99                     }
100                 }
101             }
102         } ]
103     } ]

```

- **Physical and logical locations of the vulnerability**

This examples shows data for physical and logical locations of a vulnerability:

The data for execution steps in this section includes a code snippet and rendered data from intermediate representation (IR) data (user code is not usually displayed). Contrast uses this data to analyze what the scan sees.

1. This example shows the execution statement from IR.
2. This example shows the general area where an execution step occurs in the application.


```

44     "threadFlows" : [ {
45         "locations" : [ {
46             1 "location" : {
47                 "physicalLocation" : {
48                     "artifactLocation" : {
49                         "uri" : "com/scalesec/vulnado/LoginController.java"
50                     },
51                     "region" : {
52                         "startLine" : 19,
53                         "snippet" : {
54                             "rendered" : {
55                                 "text" : "@CrossOrigin(origins={\\\"*\\\"})\\n@RequestMapping(value=\\\"/login\\\",
56                                     method=\\\"POST\\\", produces=\\\"application/json\\\", consumes=\\\"application/
57                                     json\\\")\\nLoginResponse login(**@RequestBody LoginRequest input**) {\\n ...\\n\"
58                                 }
59                             },
60                             "properties" : {
61                                 "ir" : [ "@CrossOrigin(origins={\\\"*\\\"})\\n@RequestMapping(value=\\\"/login\\\",
62                                     method=\\\"POST\\\", produces=\\\"application/json\\\", consumes=\\\"application/json\\\"
63                                     )\\nLoginResponse login(**@RequestBody LoginRequest input**) {\", \" ...\", \"}\" ]
64                             }
65                         }
66                     },
67                     "logicalLocations" : [ {
68                         "name" : "LoginController.login()",
69                         "fullyQualifiedName" : "com.scalesec.vulnado.LoginController.login(com.scalesec.
70                         vulnado.LoginRequest)"
71                     } ]
72                 }
73             }
74         ]
75     }

```

• Untrusted data location

These examples shows scan results for untrusted data.

Scan looks at executions steps from code source to data sink. The scan results include any execution step that touches untrusted data.

If the importance result is essential, check this area of the code for vulnerabilities.

1. This example shows the location of tainted data that the scan is tracking.
2. This example shows the location of the tracked data that the execution step touches. The scan results indicate the importance is essential. This code needs to be checked for vulnerabilities.

```

"location" : {
  "physicalLocation" : {
    "artifactLocation" : {
      "uri" : "com/scalesec/vulnado/User.java"
    },
    "region" : {
      "startLine" : 47,
      "snippet" : {
        "rendered" : {
          "text" : "public static User fetch(String un) {\n ... \n $stack0 = new\nStringBuilder(); // Java line 47\n $stack1 = \"select * from users where\nusername = '\""; // Java line 47\n $stack0 = $stack0.append($stack1); //\nJava line 47\n $stack0 = $stack0.append(un); // Java line 47\n $stack1 =\n\"' limit 1\"; // Java line 47\n $stack0 = $stack0.append($stack1); //\nJava line 47\n query = $stack0.toString(); // Java line 47\n ... \n}"
        }
      }
    },
    "message" : {
      "text" : "Propagation event of untrusted data occurred at 'User.java:47' in the\nmethod 'User.fetch()' to variable 'query'"
    },
    "properties" : {
      "ir" : [ "$stack0 = new java.lang.StringBuilder; // Java line 47", "$stack1 =\n\"select * from users where username = '\""; // Java line 47", "$stack0 =\nvirtualinvoke $stack0.<java.lang.StringBuilder: java.lang.StringBuilder append\n(java.lang.String)>($stack1); // Java line 47", "$stack0 = virtualinvoke\n$stack0.<java.lang.StringBuilder: java.lang.StringBuilder append(java.lang.String)>\n(un); // Java line 47", "$stack1 = '\"' limit 1\"; // Java line 47", "$stack0\n= virtualinvoke $stack0.<java.lang.StringBuilder: java.lang.StringBuilder append\n(java.lang.String)>($stack1); // Java line 47", "query = virtualinvoke $stack0.\n<java.lang.StringBuilder: java.lang.String toString>(); // Java line 47" ]
    }
  },
  "logicalLocations" : [ {
    "name" : "User.fetch()",
    "fullyQualifiedName" : "com.scalesec.vulnado.User.fetch(java.lang.String)"
  } ],
  "message" : {
    "text" : "query"
  }
},
"state" : {
  "query" : {
    "text" : "tainted",
    "properties" : {
      "taintTags" : [ "untrusted", "cross-site" ]
    }
  }
},
"importance" : "essential"

```

Scan analysis

The examples in this section show how to identify content in the SARIF file that can help you analyze scan results.

- **Classes that scan uses to trace data**

These classes affect scan execution time.

```

2344 "scannedData" : {
2345   "scannedBodyClasses" : [ "com.scalesec.vulnado.ServerError",
2346     "com.scalesec.vulnado.LoginResponse",
2347     "org.springframework.lang.UsesSunMisc",
2348     "com.scalesec.vulnado.Postgres",
2349     "com.scalesec.vulnado.LinksController",
2350     "com.scalesec.vulnado.BadRequest",
2351     "com.scalesec.vulnado.Unauthorized",
2352     "com.scalesec.vulnado.Comment",
2353     "org.springframework.lang.NonNullFields",
2354     "org.springframework.lang.NonNull",
2355     "org.springframework.lang.UsesJava7",
2356     "org.springframework.lang.UsesJava8",
2357     "com.scalesec.vulnado.CowController",
2358     "com.scalesec.vulnado.LoginController",
2359     "org.springframework.lang.NonNullApi",
2360     "org.springframework.lang.Nullable",
2361     "org.springframework.lang.UsesSunHttpServer",
2362     "com.scalesec.vulnado.CommentsController",
2363     "com.scalesec.vulnado.Cowsay",
2364     "com.scalesec.vulnado.VulnadoApplication",
2365     "com.scalesec.vulnado.LoginRequest",
2366     "com.scalesec.vulnado.LinkLister",
2367     "com.scalesec.vulnado.User",
2368     "java.util.Optional",
2369     "com.scalesec.vulnado.CommentRequest" ],

```

- **Classes used for type hierarchy resolution**

Scan uses these classes only for type hierarchy resolution.

The library classes are either not relevant to security issues or Contrast has a specific policy for the relevant API.

If a class displayed in this section is related to custom code, it is possible that the scan results contain false negatives.

```

2134 "nonScannedBodyClasses" : [
2135     "java.nio.file.WatchEvent$Modifier",
2136     "java.awt.Color", "java.awt.peer.WindowPeer",
2137     "java.util.function.IntUnaryOperator",
2138     "sun.awt.datatransfer.DataTransferer",
2139     "java.awt.JobAttributes$MultipleDocumentHandlingType",
2140     "java.lang.Integer",
2141     "java.awt.image.SampleModel",
2142     "javax.swing.border.Border",
2143     "java.awt.peer.ScrollbarPeer",
2144     "java.util.Vector",
2145     "java.sql.DriverAction",
2146     "java.sql.SQLType",
2147     "org.springframework.core.ParameterizedTypeReference$1",
2148     "java.nio.file.Path",
2149     "java.nio.channels.Channel",
2150     "org.springframework.core.io.Resource",
2151     "java.awt.peer.ContainerPeer",
2152     "javax.swing.KeyStroke",
2153     "java.lang.CharSequence",
2154     "java.awt.dnd.DropTargetDragEvent",
2155     "java.time.temporal.TemporalField",
2156     "org.springframework.beans.factory.InjectionPoint",
2157     "java.util.logging.ErrorManager",
2158     "java.awt.Scrollbar",
2159     "java.io.Serializable",
2160     "java.util.concurrent.CompletionStage",
2161     "java.lang.LayerInstantiationException",
2162     "org.jsoup.parser.Token$EndTag",
2163     "java.lang.invoke.BoundMethodHandle$Specializer$Factory",
2164     "java.lang.invoke.MemberName",
2165     "java.util.function.ToDoubleFunction",
2166     "java.io.ObjectInputStream$GetField",

```

- **Phantom classes**

Phantom classes are referenced classes but either Scan is unable to find bytecode for them or Scan was unable to decompile the code into IR.

Ideally, the scan results should contain no phantom classes. If the results include phantom classes, Scan was unable to find the data it needed to provide more accurate results. If you see application code or libraries displayed in this section, look at your code to determine if an issue exists.

```

2400 "phantomClasses" : [ "BOOT-INF.classes.com.scalesec.vulnado.LoginController",
2401     "javax.annotation.Nonnull",
2402     "groovy.lang.Closure",
2403     "javax.annotation.meta.TypeQualifierDefault",

```

- **Discovered routes for untrusted data**

This example shows discovered routes where untrusted data enter the application.

Scan only looks at the data flow behavior from the functions displayed in this section. Scan does not analyze other functions related to data flow, such as weak cryptography.

```

2769     "routesDiscovered" : [ {
2770         "routeSignature" : "com.scalesec.vulnado.LoginController.login(com.scalesec.vulnado.
LoginRequest)"
2771     }, {
2772         "routeSignature" : "com.scalesec.vulnado.CowController.cowsay(java.lang.String)"
2773     }, {
2774         "routeSignature" : "com.scalesec.vulnado.CowController.cowsay2(java.lang.String)"
2775     }, {
2776         "routeSignature" : "com.scalesec.vulnado.LinksController.links(java.lang.String)"
2777     }, {
2778         "routeSignature" : "com.scalesec.vulnado.LinksController.linksV2(java.lang.String)"
2779     }, {
2780         "routeSignature" : "com.scalesec.vulnado.CommentsController.comments(java.lang.String)"
2781     }, {
2782         "routeSignature" : "com.scalesec.vulnado.CommentsController.createComment(java.lang.String,com.
scalesec.vulnado.CommentRequest)"
2783     }, {
2784         "routeSignature" : "com.scalesec.vulnado.CommentsController.deleteComment(java.lang.String,
java.lang.String)"
2785     } ]
2786 },
2787 "invocations" : [ {
2788     "commandLine" : "java -XX:MaxRAMPercentage=80 -jar /app/contrast-scan-java-cli.jar
--prescan-metadata /tmp/
cb9757b4-4fdf-4de9-bdf1-7769058307eb_e1a6403d-be7c-46ee-82aa-6b7e47ce97d2_metadata.json -o /tmp/
results.sarif.json /tmp/
cb9757b4-4fdf-4de9-bdf1-7769058307eb_e1a6403d-be7c-46ee-82aa-6b7e47ce97d2_vulnado-0.0.1-SNAPSHOT.
jar",
2789     "toolExecutionNotifications" : [ ],
2790     "executionSuccessful" : true

```

View scan policies

Policies determine which vulnerabilities Contrast looks for in your code. At this time, editing or adding policies is not supported.



NOTE

The policy view applies to Java binary scanning only.

1. Select **Scans** in the header.
2. Select a scan project.
3. Select **Policy**.
The policy list displays the Java binary scan policies.
4. In the Policy tab, search for a specific policy by entering one or more characters in the Find box.

Change scan settings

Scan settings let you change the name of a scan project and update metadata.

You can also:

- [Archive scan projects \(page 906\)](#)
- [Delete scan projects \(page 667\)](#)
- [Configure dynamic scoring \(page 664\)](#)

Before you begin

- If role-based access is turned on, you need a role with the View, edit, delete project action to update all settings except dynamic scoring.

For the dynamic scoring option, you need a role with the Manage organization action.

- If you are using organization users and groups, you need an Organization Admin role.

Steps

1. Select **Scans** in the header.
2. Select a Scan project.
3. Select the **Settings** icon (⚙️) at the top of the list.
4. Enter a new name for the project.
5. Update metadata values for the scan project.
If metadata values are configured, you can change the displayed values. If no metadata values are configured, you must [create them \(page 663\)](#) in the Scan projects settings for the organization.
6. Select **Save**.

Archive scan projects

If you want to exclude a specific scan project from the list of scan projects (for example, you are no longer using that project), you can archive it.

Contrast keeps the data associated with the archived project.

Steps

1. Select **Scans** in the header.
2. Select a scan project.
3. Select the settings icon (⚙️).
4. In the Scan project settings window, select **Archive**.



5. In the Archive project window, select **Archive**.
The archived project is no longer visible in the list unless you use the Archived filter.

Unarchive scan projects

To use or view details for an archived scan project, unarchive it.

Before you begin

- An Organization Admin role is required.

Steps

1. Select **Scans** in the header.
2. Display archived projects by selecting the small triangle (▾) at the top of the list and then, selecting **Archived**.

3. Hover over the end of a project row and select the **Unarchive** icon (🗑️).
4. In the Unarchive project window, select **Unarchive**.

Integrate scans with build pipelines

The [Contrast CLI \(page 1002\)](#) has commands that let you run a scan without using the Contrast web interface.

This topic provides instructions for using the Contrast CLI to integrate scans into any build pipeline.

You can also use the [Contrast Maven plugin \(page 1103\)](#) to integrate Contrast Scan into your project's Maven build,

Before you begin

- In the Contrast web interface, under **user menu > User settings > Profile**, locate and copy this information:
 - API key
 - Organization ID
 - Contrast URL
 - Authorization header
- Ensure that a WAR or JAR file is available in an accessible location.

Steps

1. In your build pipeline workflow, add the command to download the latest version of the Contrast CLI.

```
npm install --location=global @contrast/contrast@2
```

2. Set environment variables for the API key, the organization ID, the Contrast URL, and the Authorization header.

This example shows how to set the environment variables with GitHub secrets. Use the appropriate method for your environment.

```
CT_API_KEY: ${ secrets.CONTRAST__API__API_KEY }
CT_AUTH_TOKEN: ${ secrets.CONTRAST__API__AUTH_TOKEN }
ORG_ID: ${ secrets.CONTRAST__API__ORGANIZATION_ID }
URL: ${ secrets.CONTRAST__API__URL }
```

3. Add a command similar to the following to start each scan:

```
contrast --scan ../scan-cli-testing/java/apps/param.war \
--api_key $CT_API_KEY \
--authorization $CT_AUTH_TOKEN \
--organization_id $ORG_ID \
--host $URL \
--project_name MY-Project \
--language JAVA --wait_for_scan
```

When this command runs for the first time, Scan creates a project using the name specified in the `--project_name` option.

The output from the command looks similar to this example:

```
project created ID is 788f9734-b933-4f05-b391-c130931baf88
Uploaded file successfully.
Response: {
  id: '5091d134-93ea-4873-8110-8cf99d14606e',
```

```
organizationId: '74f4cd04-6ca9-4eb7-a7a7-78909c2101cc',
projectId: '788f9734-b933-4f05-b391-c130931baf88',
filename: 'param.war',
createdTime: '2022-04-04T10:06:16.952+00:00'
}
Timeout set to 5 minutes
Waiting for results...
New Results: 5
Fixed Results: 0
Total Results: 5
```

The next time you run the command for the same project, Scan adds the uploaded files to the original project. The output from the command looks similar to this example:

```
project already exists with this name. Getting ID...
project ID is 788f9734-b933-4f05-b391-c130931baf88
Uploaded file successfully.
Response: {
  id: '94b4e065-0e0f-46bb-b1d8-9f85bd03c602',
  organizationId: '74f4cd04-6ca9-4eb7-a7a7-78909c2101cc',
  projectId: '788f9734-b933-4f05-b391-c130931baf88',
  filename: 'param.war',
  createdTime: '2022-04-04T10:07:01.230+00:00'
}
Timeout set to 5 minutes
Waiting for results...
New Results: 5
Fixed Results: 0
Total Results: 5
```

4. After the scan completes, go to the Contrast web interface to view the Scan project details and results.

The screenshot shows the Contrast web interface. The top navigation bar includes 'Contrast', 'Applications', 'Scans' (highlighted with a red box), 'Servers', 'Libraries', 'Vulnerabilities', and 'Attacks'. A search bar and a notification bell with '21' are also present. The main content area is for 'MY-Project' (Language: Java, Last Scan: 12 Minutes Ago). It features tabs for 'Overview', 'Vulnerabilities', and 'Policy'. A '+ New Scan' button is in the top right. The overview section displays a progress bar for '0/100' and several metrics: 37 Vulnerabilities, 0 New Vulnerabilities, 0 Remediated, 2 Scans Completed, and 0 Days Since Last Scan. Below this is the 'SCAN HISTORY' section, showing a table of scans. The table has columns for 'Vulnerabilities', 'Label', 'Scan Date', 'Language', and 'Coverage'. Two scans are listed, both with 37 vulnerabilities and a 'View' link. A pagination bar at the bottom shows '1' selected and 'Results per page 10'.

Vulnerabilities	Label	Scan Date	Language	Coverage
37	2022-04-29 19:28:44	13 minutes ago	Java	View
37	2022-04-29 19:20:51	21 minutes ago	Java	View

Examples

- [Scan integration with GitHub \(page 909\)](#)
- [Scan integration with Jenkins \(page 909\)](#)
- [Scan integration with GitLab \(page 910\)](#)

Example: Scan integration with GitHub

Review the [Scan integration steps \(page 907\)](#) before you integrate Contrast Scan with GitHub.

This example shows how to set up a GitHub workflow.

```
- name: Set up contrast-cli
  run: |
    npm install --location=global @contrast/contrast@2.0.0
- name: Scan file
  env:
    CT_API_KEY: ${ secrets.CONTRAST__API__API_KEY }
    CT_AUTH_TOKEN: ${ secrets.CONTRAST__API__AUTH_TOKEN }
    ORG_ID: ${ secrets.CONTRAST__API__ORGANIZATION_ID }
    URL: ${ secrets.CONTRAST__API__URL }
  run: |
    contrast-cli --scan ./target/${ inputs.SERVICE_NAME }-${ steps.build-
service.outputs.version }.jar \
      --api_key $CT_API_KEY \
      --authorization $CT_AUTH_TOKEN \
      --organization_id $ORG_ID \
      --host $URL \
      --project_name MY-Project \
      --language JAVA --wait_for_scan
```

Example: Scan integration with Jenkins

Review the [Scan integration steps \(page 907\)](#) before you integrate Scan with Jenkins.

Contrast Security can share this script to integrate scans with a Jenkins pipeline ([contact Contrast Support](#) to access these scripts):

- **Jenkins Pipelines script:** The `Jenkins_Script_SCAN` script uses the Contrast Scan local engine JAR file. The project JAR file is expected to be in a GitHub repository.

Integration setup

This example describes how to set up a Jenkins integration for Scan.

1. Set up a Jenkins instance in your local environment (use the [Jenkins documentation](#)). If you already have a Jenkins instance, you can skip this step.
2. Install this software (if not already installed):
 - Java 11
 - Plugins for your environment (if needed)
3. Create a new pipeline and copy the Contrast script.
4. Set the Contrast credentials as global or environment variables:
For example: `URL`, `USER_NAME`, `API_KEY`, `SERVER_KEY`, `ORGANIZATION`.
 - To add credential to Jenkins, select **Manage Jenkins > Manage Credentials > Add Credentials as Secret Text**.
5. Refer to all credentials and variables in your pipeline scripts.

Example: Scan integration with GitLab

Review the [Scan integration steps \(page 907\)](#) before you integrate Scan with GitLab.

This example shows how to set up a GitLab pipeline for these actions:

- Pulling code from a repository.
This action might not be necessary if you're using GitLab as a code repository.
- Building the code
- Scanning a generated JAR file.

Pipeline setup example

This sample YAML file shows the steps to set up the GitLab pipeline.

```
stages:                # List of stages for jobs, and their order of execution
  - pull
  - build
  - scan
#  - deploy

pull:
  stage: pull
  artifacts:
    paths:
      - WebGoat
  script:
    - git clone -b main https://github.com/WebGoat/WebGoat.git

build:
  stage: build
  image: maven:3.8.1-openjdk-17-slim
  artifacts:
    paths:
      - $CI_PROJECT_DIR
  dependencies:
    - pull
  script:
    - ls -l /tmp
    - cd WebGoat
    - mvn -DskipTests clean install

scan:                  # This is the step for Contrast Scan
  stage: scan
  image: node:18.19-slim
  dependencies:
    - build
  script:
    - ls -la
    - npm install -g @contrast/contrast@2
    - contrast version
    - contrast auth --api-key $API_KEY --authorization $AUTH --organization-
id $ORG_ID --host $URL
    - contrast scan -f $CI_PROJECT_DIR/WebGoat/target/webgoat-2023.7.jar --
fail --severity high

deploy: #TODO
```

Servers

In Contrast, you can see servers and configure how they function in development, test (QA), and production environments. You are then able to compare the differences across environments as code travels. Contrast sets up a shell for you to designate servers. Once that's in place, Contrast can begin to find weaknesses.

Server settings

Each server entry in Contrast represents a Contrast agent that you installed for an application. Contrast creates a new, unique server entry when you configure these settings for each agent:

To define custom settings for servers, use these entries in the Contrast configuration file:

- **Server name:** The default value is the host name.
- **Server path:** The path from which the agent process is running.
- **Server type:** The type of server hosting your application.

Using custom values for these settings instead of default ones is useful if you want to avoid duplicate server entries.

Configure the server environment with this setting:

- **Server environment:** The environment in which you want to use this server.
The valid values are `DEVELOPMENT`, `QA`, and `PRODUCTION`. These values are case-insensitive. The default value is `DEVELOPMENT`.

Contrast agents automatically recognize all supported server types. If the server type is not automatically recognized, it may be due to the agent running with an unsupported technology. Please review the *Supported Technologies* section for the respective [agent \(page 51\)](#). Running in unsupported environments may affect the functionality of some features like route discovery.

Settings in a configuration file

These are the server settings that you customize in a configuration file.

```
# server
# Use the settings in this section to set
# metadata for the server hosting this agent.

server:

  # Override the reported server name.
  name: localhost

  # Override the reported server path.
  path: NEEDS_TO_BE_SET

  # Override the reported server type.
  type: NEEDS_TO_BE_SET

  # Override the reported server environment.
  # environment: DEVELOPMENT
```

Agent configuration instructions

When you define custom settings for servers, use the configuration instructions for the agent you are using:

- [.NET Core configuration \(page 290\)](#)

- [.NET Framework configuration \(page 228\)](#)
- [Go configuration \(page 534\)](#)
- [Java configuration \(page 149\)](#)
- [Node.js configuration \(page 352\)](#)
- [Python configuration \(page 413\)](#)
- [Ruby configuration \(page 473\)](#)

Contrast options

The Contrast web interface provides additional options for server configuration:

- [Configure a server \(page 915\)](#)
- [Output data to syslog \(page 918\)](#)

View servers

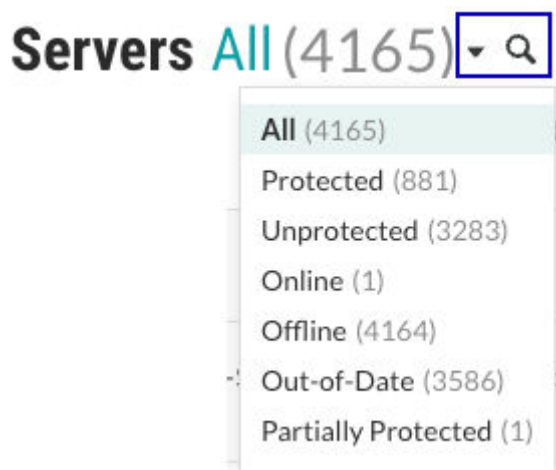
The Servers list shows details about the servers in your organization:

- **Name:** The name of the server.
- **First Seen:** The first time the agents for the applications associated with the server reported activity to Contrast.
- **Last Seen:** The last time the agents for the applications associated with the server reported activity to Contrast.
- **Environment:** The environment where the server is deployed: Development, QA, or Production.
- **Applications:** The applications associated with the server.

The Servers list also lets you manage Assess and Protect settings for individual servers.

Steps

1. Select **Servers** in the header to view a list of all servers in your organization.
2. To filter the list by server status, select the small triangle (▾) at the top of the list.
Alternatively, search for specific servers by name by selecting the magnifying glass icon (🔍).

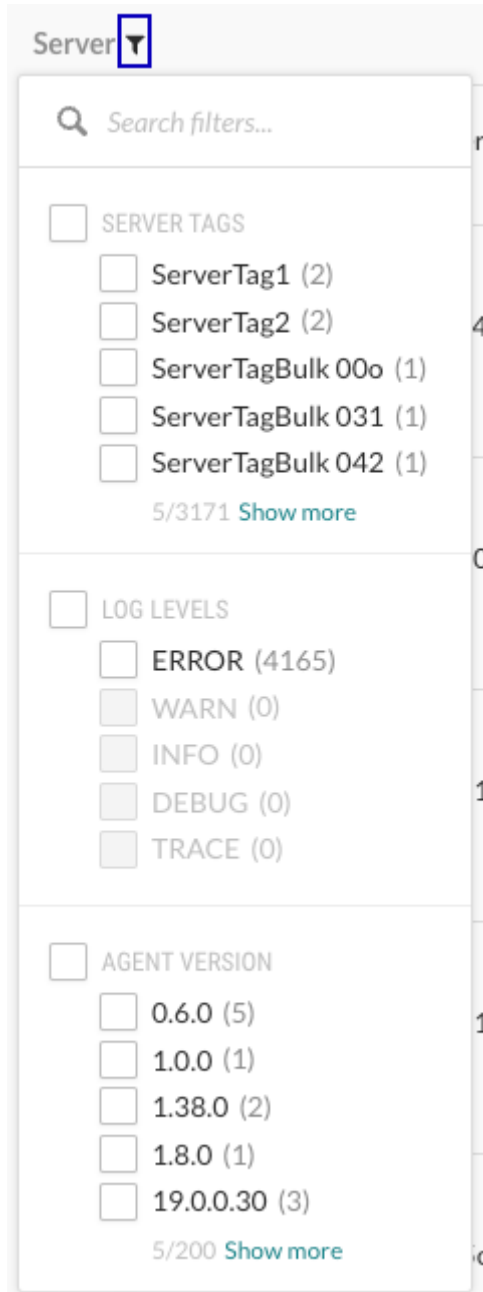


The filters are:

- **All:** Shows all servers in the organization.
- **Protected:** Servers with Protect turned on.
- **Unprotected:** Servers with Protect turned off.
- **Online:** Servers that Contrast can reach.
- **Offline:** Servers that Contrast is unable to reach.

- **Out-of-Date:** The agent version for applications associated with the server is old. Consider updating the agent to a newer version.
- **Partially protected:** Servers with Protect turned on but require a restart.
If you change the agent configuration for an application associated with the server, you might need to restart it for Protect to take effect.

3. To filter the list of servers, select the filter icon (▼) next to the Server column header.



The filters are:

- **Server tags:** Tags you assigned for each server.
To assign a tag to a server, hover over the end of a row for a server and select the Tag (◆) icon.
- **Log levels:** [Log levels \(page 1271\)](#) assigned to each server.
- **Agent version:** The version of the agents that the applications associated with the server are using.

4. To filter by environment, select the filter icon (▼) next to the Environment column header.
The filters are: Development, QA, and Production.

5. Use the settings in the [Assess \(page 1153\)](#) and [Protect \(page 1155\)](#) columns to turn them on or off.

- If you use only the Contrast web interface to turn Assess or Protect on or off, the setting for a specific server is green if ON and gray if OFF. You can change this setting in the Contrast web interface.



- If you used a method external to the Contrast web interface to configure the setting for Assess or Protect (for example, an agent configuration file), the setting is green but disabled if ON and grey but disabled if OFF. You cannot change this setting in the Contrast web interface.



- If the setting in the Contrast web interface is disabled, hover over the setting to see where it is configured. The [order of precedence \(page 106\)](#) determines which setting Contrast uses as the effective configuration.

6. To view [details about a specific server \(page 914\)](#), select the server name.

Server details

When you select a server from the Servers list and view the Overview tab, you see details about the server configuration and activity for applications associated with the server.

You can also manage the settings for Protect and Assess.

Summary

The Summary at the top of the Overview tab shows these values:

• Assess and Protect settings

You can use configuration files, variables, or the Contrast web interface to configure the Protect and Assess settings. The method you use to configure these settings determines whether you can change them in the Contrast web interface.

- If you used only the Contrast web interface to turn Protect or Assess on or off, the setting is green if ON and gray if OFF. You can change this setting in the Contrast web interface.



- If you used a method external to the Contrast web interface to configure the setting for Protect or Assess (for example, an agent configuration file), the setting is green but disabled if ON and grey but disabled if OFF. You cannot change this setting in the Contrast web interface.



If the setting in the Contrast web interface is disabled, hover over it to see where the setting is configured. The [order of precedence \(page 106\)](#) determines which setting Contrast uses as the effective configuration.

- **Agent version:** The version of the agent associated with this server.
- **Libraries:** The number of open-source libraries that Contrast identifies for the applications associated with the selected server. It also displays the number of vulnerable libraries.

Select the displayed number to view the libraries list.

- **Time since last startup:** The amount of time since the server last started.
- **Since last seen:** Time since the agent associated with this server reported activity to Contrast.

Statistics

The statistics section shows these values:

- **Vulnerabilities:** If Assess is turned on, the number of vulnerabilities it identified for the applications associated with this server.
Use the filter to change the view. Hover over the vulnerability bar to see additional details.
- **Attacks:** If Protect is turned on, the number of attacks identified in the applications for this server.
Use the filter to change the view. Hover over the attacks bar to see additional details.
- **Application:** The applications associated with this server.
Select an application link to see application details.

Activity

The activity graph shows an aggregate of agent reports that Contrast received during the selected time frame.

Configure server settings

Server settings let you configure how a server functions in each environment (development, test, and production).

Steps

1. Select **Servers** in the header.
2. Find the server you want to modify using either of these methods:
 - Select the Filter icon (▼) at the top of the Server column.
 - Use the magnifying glass (🔍) to search.
3. Go to Server settings using either of these methods:
 - Hover over the end of the server's row and select the **Settings** icon (⚙️).
 - Select the name of the server and then, select the **Settings** icon (⚙️) at the top of the list.
4. Modify the settings, as needed:
 - Modify the server name.
 - Designate the environment in which the server will be running: Development, QA (test), or Production.
 - In the Server log file, override the existing server log file path by entering the preferred path.



NOTE

Server log files are restricted to file types of LOG or TXT only.

- Set the [log level \(page 1271\)](#) for the server.
 - Set bot blocking.
Bot blocking blocks traffic from scrapers, attack tools, and other unwanted automation.
To view blocked bot activity, under **Attacks > Attack Events**, use the **Automated** filter options.
- Supported languages:** Java, .NET Framework, .NET Core, Ruby, and Python.



NOTE

You can configure bot blocking in the [YAML files \(page 107\)](#) for Java, .NET Framework, .NET Core, Ruby, and Python.

- Select **Enable sampling for higher performance (page 916)**.

This setting is available when Assess is enabled.

Configure the following settings:

- **Baseline:** The number of times that Contrast analyzes URLs to complete sampling. The default setting is **5**.
- **Frequency:** The number of times that Contrast analyzes URLs after the baseline is achieved. The default setting is **10**.
- **Window:** The number of seconds that Contrast retains samples before reverting to the baseline. The default setting is **180**.

For example:

```
contrast.assess.sampling.request_frequency    25
contrast.assess.sampling.window_ms           360_000
contrast.assess.sampling.baseline            1
```

- Select **Enable output of Protect events to syslog. (page 918)**

This setting is available when Protect is enabled.

Select the syslog message severity levels that the server outputs to syslog. Contrast offers syslog message categories according to the [syslog RFC 3164](#) specification for severity.

Application sampling

If you notice performance issues after you instrument your application with a Contrast agent, consider enabling application sampling.

When enabled, sampling selectively turns the agent off for a short amount of time, based on uniquely identified requests. If an application is responding to the same request often, the agent doesn't need to analyze it multiple times. This behavior ensures that the agent analyzes code only when it's in different contexts.

Best practice: Consider enabling sampling when you run long-duration tests in a test environment. Due to the high-level of activity that occurs in this situation, enabling sampling can improve application performance.

If sampling is appropriate for your environment, enable it as part of your [server configuration \(page 915\)](#).

How sampling works

1. If the Contrast agent sees the same URL being called multiple times, it analyzes the URL based on the the number of times specified in the Baseline setting.
2. Afterwards, if the Contrast agent continues to see the same URL, it only checks the URL based on the Frequency setting.
3. Contrast retains samples for the number of seconds specified for the Window setting. After the time specified for the Window setting elapses, Contrast analyzes the URL again, according to the Baseline setting.

Use automatic diagnostic collection

Automatic diagnostic collection lets Contrast collect diagnostic information without requiring you to find or upload this information manually. Automatic collection makes it easier for you to work with Contrast support when you are troubleshooting technical issues.

**NOTE**

Currently, this feature is in preview mode. Contrast Support contacts you if they need this feature enabled during troubleshooting activity.

When you turn on this feature, Contrast collects logs, system data, and other diagnostic information from Contrast agents and sends them to Contrast servers. Contrast support representatives access this information during troubleshooting tasks.

Before you begin

- Currently, only Java agents support automatic diagnostic collection.
- The time span for diagnostic collection is one to 25 hours, based on the value you specify.

Steps

1. Select **Servers** in the header.
2. Hover over the end of the row of a server and select the diagnostic (🔍) icon.
3. In Diagnostic collection, select the log level for the logs you want Contrast support to access:
 - **Trace logs:** Includes the trace messages generated during one or more sessions.
 - **Debug logs:** Includes information that helps to identify bugs or other issues.

Diagnostic Collection

Turning on Diagnostic Collection automatically sends diagnostic data from Contrast agents to Contrast Support staff and servers. It removes the need for you to gather logs, memory dumps, or other information manually. Read the disclaimer before you turn on Diagnostic Collection.

Send **DEBUG** logs for next **2** hours.

! Disclaimer

By opting-in to the diagnostic collection, you agree to allow Contrast Security access to your diagnostic information. This includes, but is not limited to: logs, data dumps, memory dumps, configuration information, and other sources of sensitive Personal Identifiable Information (PII) or secrets. Turning on diagnostic collection instructs Contrast to send diagnostic information for the specified amount of hours. This feature might not work for on-premise customers.

Cancel

Enable Diagnostic Collection

4. Specify the collection time span, from one to 25 hours. Contrast support can advise you regarding the amount of time required for troubleshooting tasks.
5. Select **Enable diagnostic collection**.
The color of the diagnostic icon changes to green (🟢) when you hover over the end of the server row.
6. Select the diagnostic icon (🟢) and copy the displayed key in Diagnostic Collection.. Give this key to your Customer support representative.

Diagnostic Collection

Turning on Diagnostic Collection automatically sends diagnostic data from Contrast agents to Contrast Support staff and servers. It removes the need for you to gather logs, memory dumps, or other information manually. Read the disclaimer before you turn on Diagnostic Collection.

Log Level	Collection Start	Collection End
DEBUG	Invalid Date Invalid Date	Invalid Date Invalid Date

Don't want diagnostics? ☒ Stop Data Collection

Key: fab3cd29-59d5-4b1c-8086-989a27b60ba2

Disclaimer

By opting-in to the diagnostic collection, you agree to allow Contrast Security access to your diagnostic information. This includes, but is not limited to: logs, data dumps, memory dumps, configuration information, and other sources of sensitive Personal Identifiable Information (PII) or secrets.. Turning on diagnostic collection instructs Contrast to send diagnostic information for the specified amount of hours. This feature might not work for on-premise customers.

Cancel

7. To turn off diagnostic collection, repeat steps 1 and 2. Then, select **Stop diagnostic collection..**

Diagnostic Collection

Turning on Diagnostic Collection automatically sends diagnostic data from Contrast agents to Contrast Support staff and servers. It removes the need for you to gather logs, memory dumps, or other information manually. Read the disclaimer before you turn on Diagnostic Collection.

Log Level	Collection Start	Collection End
DEBUG	Invalid Date Invalid Date	Invalid Date Invalid Date

Don't want diagnostics? ☒ Stop Data Collection

Key: fab3cd29-59d5-4b1c-8086-989a27b60ba2

Disclaimer

By opting-in to the diagnostic collection, you agree to allow Contrast Security access to your diagnostic information. This includes, but is not limited to: logs, data dumps, memory dumps, configuration information, and other sources of sensitive Personal Identifiable Information (PII) or secrets.. Turning on diagnostic collection instructs Contrast to send diagnostic information for the specified amount of hours. This feature might not work for on-premise customers.

Cancel

Send output to syslog

Contrast lets you send security logs to a remote syslog server in addition to the Contrast Security log. [Syslog message format \(page 921\)](#) describes the components in the messages that Contrast sends.

Before you begin

- You must apply a Protect license to the server that has syslog output enabled.
- You may have to enable remote logging so that your syslog can receive outside messages.
- Syslog messages for a server are sent by the agent.
- Syslog output isn't supported over TCP.

Steps

1. When configuring the [default organization server settings \(page 1171\)](#), select the checkbox to **Enable output of Protect events to syslog**, which reveals additional fields, and then enter the appropriate settings.
2. Select **Servers** in the header to enable and configure syslog output to an individual server or multiple servers at one time. If syslog defaults have already been [set at an organization level \(page 1171\)](#), the values will be pre-populated for server-level settings.
 - **Individual server:** To enable syslog on an individual server, hover over the grid row, and select the **Server settings** icon.
 - **Multiple servers:** Use the check marks to select multiple servers, and select the **Server Settings** icon in the batch action menu that appears at the bottom of the page.



NOTE

If one or more of the selected servers is not eligible to have syslog enabled, it will only be enabled on eligible servers.

3. In the **Server settings** window, select the box to **Send output of Protect events to syslog**. (For multiple servers, you will need to select **Edit** next to the checkbox first).



NOTE

If eligible servers selected are in different environments, you can choose to use the default settings for the applicable servers or manually configure the settings for all servers.

Bulk Server Settings



This will override settings for all relevant selected servers.

Environment

Log Level

Stacktraces

Bot Blocking

☐ [Edit](#)☐ Enable sampling for higher performance [?](#) [Edit](#)☒ Enable output of Protect events to syslog [EDITED](#)

Syslog Server Host

Port

Facility

Attack Event Result

EXPLOITED

SUSPICIOUS

BLOCKED

BLOCKED (P)

PROBED

Syslog Message Severity

[Undo edits](#)[Cancel](#)[Save](#)

4. Enter the **Syslog server host**. This can be the full qualified domain name (not just the hostname) or the IP address. For example: *email.mydomainname.com* or 38.124.154.50.
5. Enter the **Port**.

6. Enter the **Facility**.
7. Enter the **Syslog message severity**.
8. **Save** the settings to enable syslog on the server.
9. When syslog is enabled, the server has a gray arrow icon beside its name in the grid. Hover over the icon to see the output location of Protect events.
To edit server settings, repeat the steps above to update the values in the appropriate configuration form, and save your changes.

Syslog message format

Syslog messages are formatted using the Common Event Format (CEF), a standard format that most Security Information and Event Management (SIEM) solutions use to facilitate the parsing and analysis of log data.

The following example shows the format that Contrast uses for syslog messages that it sends:

```
Aug 04 2024 192.168.219.65 CEF:0|Contrast Security|Contrast
Agent Java|4.5.2.0|SECURITY|The parameter undefined2 had
a value that was marked suspicious reflected -xss
- <script>alert('TURTLES-everywhere')</script>|WARN|pri=reflected-xss
src=10.80.49.96 spt=8443 request=/actuator/info requestMethod=GET
app=svc-basic outcome=SUSPICIOUS dvchost=my.example.hostname.com
computer=my.example.hostname.com contrastAgentServer=my-server-name from
contrast.yaml
```

The components in the Contrast syslog message are:

- **Date:** The timestamp of the local host indicating when the log was generated. For example: Aug 04 2024.
The timestamp uses this format: MMM dd HH:mm:ss, which shows the current time zone of the server running the agent.
- **Host:** The host address where the log was generated. For example: 192.168.219.65.
- **CEF Version:** The version of the CEF format being used. For example: CEF:0.
- **Device Vendor:** The vendor of the security device. For example: Contrast Security.
- **Device Product:** The product that is generating the log. For example: Contrast Agent Java.
- **Device Version:** The version of the product generating the log. For example: 4.5.2.0 .
- **Signature ID:** The type of event. For example, SECURITY indicates a security-related event.
- **Message:** A description of the event. For example: The parameter undefined2 had a value that was marked suspicious reflected -xss -<script>alert('TURTLES-everywhere')</script>.
- **Log level:** The log level of the event. For example, WARN or DEBUG.
- **Extensions:** Other fields that provide further details about the event. Some examples include:
 - **pri:** Priority of the event.
 - **src:** Source IP address of the event.
 - **spt:** Source port.
 - **request:** The specific request that triggered the event.
 - **requestMethod:** The HTTP method used.
 - **app:** The application involved in the event.
 - **outcome:** The outcome of the event (for example, Suspicious or Blocked).
 - **dvchost:** The hostname of the machine where the event occurred.
 - **contrastAgentServer:** The value of the `contrast.server.name` property or the host name as a fallback.

Syslog receivers

When you configure receivers for syslog output, consider these resources:

- [Contrast Protect connector for Microsoft Sentinel](#)
- [Configure Field Extraction Modules \(mmfields\) for rsyslog to parse CEF formatted logs](#)
- [Configure syslog-ng OSE for CEF formatted logs](#)

Libraries

The security of the libraries used by an application impacts the security of your application as a whole.

Libraries can be public or private. Public libraries are identified with a [score \(page 934\)](#) (A-F), public libraries are open-source libraries sourced from Maven (Java), NuGet (.NET), npm (Node.js), RubyGems (Ruby), PyPI (Python), pkg.go (Go), and Composer (PHP). Private libraries are commercial third-party libraries or custom-built libraries. Private libraries do not have a score assigned in Contrast.

Contrast agents automatically identify open-source libraries included in an application. Contrast identifies any vulnerabilities found in your libraries and confirms if the library is used at runtime.

To do this, Contrast creates a hash of the library file, which compares the file's content to a database of known library files. If the hash is in the database, Contrast can assign a [score \(page 934\)](#) to the library, provide library version information and report on the total vulnerabilities (CVEs) found in the library.



NOTE

If your library is a custom file, the hash won't be found in the database and the agent reports the library as "unknown" to the Contrast application. This may also happen if the library has recently been released or if you are using an airgap on-premises installation and have not recently [updated library definitions \(page 1215\)](#).

For Java clients, WebSphere repackages libraries at runtime, so their SHA-1 hash is different than anything known to Contrast. To preserve the SHA-1 during deployment, set the JVM system property `org.eclipse.jst.j2ee.commonarchivcore.ignore.web.fragment` to "true".

Also, any `wsadmin` calls must have the same parameter:

```
wsadmin -javaoption "-  
Dorg.eclipse.jst.j2ee.commonarchivcore.ignore.web.fragment=true  
"
```

In Contrast, select **Libraries** in the header to see an overview of all libraries across your portfolio and manage them in bulk. Select the details panel to view [CVE details \(page 928\)](#).

You can view results from a manifest (for example, package.json or pom.xml) analyzed with [Contrast CLI \(page 1003\)](#) or [scanned \(page 935\)](#) repos.

See also

[View libraries \(page 924\)](#)

[View open-source licenses \(page 933\)](#)

[Learn about library scoring \(page 934\)](#)

[CVE search \(page 938\)](#)

SCA release notes

September release

Release date: September 2023

New and improved:

- Added support for GOLANG library analysis in static SCA.
- Added a new report format (XLSX) that allows users to [export \(page 932\)](#) a two-tab report that contains the same information as the standard CSV report in the first tab and lists all libraries, each individual CVE affecting that library, the CVSS score, and the Contrast criticality in the second tab.

August release

Release date: August 2023

New and improved:

- Added the ability to filter libraries based on a specific vulnerability severity in Contrast. This can be done at both the organizational level as well as at the individual application level.
- Added a hyperlink to the CVEs entry in the NVD database to a vulnerability card. This allows an organization to see the latest information pertaining to a specific CVE.

Contrast SCA supported languages

Contrast's SCA function supports the following languages:

Runtime

Language	Sources
Java	maven central, redhat GA
.NET	Nuget, framework/core downloads
Node.js	npm
PHP	composer (packagist), Wordpress packagist
Python	pypi
Ruby	rubygems
Go	go index

CLI and SCA in repository

For further details on the supported versions for CLI see the [Contrast CLI supported languages and package managers \(page 995\)](#) page.

Language	CLI	SCA in repository
Java - Maven	✓	✓
Java - Gradle	✓	✗
.NET	✓	✗
Node	✓	✓
PHP	✓	✓

Language	CLI	SCA in repository
Python	✓	✓
Ruby	✓	✓
Go	✓	✓

View libraries

There are multiple ways to get library information:

- Select **Libraries** in the header to view a grid list of all libraries used by your organization. Select a library name from that list for more details.
- You can also see library information for an individual application or server:
 - Select **Applications** in the header, then select an application name to see its details page. Select the **Libraries** tab.
 - Select **Servers** in the header, then select a server name to see its details page. Select the **Libraries** tab.

Quick views and filters

Select the open/close filters icon ▼ to filter the libraries view.

The Quick Views filters include:

Quick Views

Select a Quick View
Vulnerable (240) ▼

Filters

Applications ▼

Tags ▼

Grades ▼

Languages ▼

Usage ▼

Licenses ▼

Environments ▼

Servers ▼

Library Severity ▼

Libraries

< Close filters ▼

☐ Score

☐ **F**

☐ **F**

☐ **F**

☐ **F**

☐ **F**

☐ **F**

☐ **F**

- **All:** Shows all libraries.
- **Vulnerable:** Shows only libraries that Contrast identified as containing CVEs.
- **Private:** Shows only commercial third-party libraries or custom-built libraries that Contrast discovered in your code.
- **Public:** Shows only the open-source libraries that Contrast discovered in your code.
- **High risk:** Shows only the libraries with a [score \(page 934\)](#) of C or below.
- **Remediated:** Shows any libraries marked as remediated.

The filters include:

- **Applications:** Find by application name.
- **Tags:** Find by tag name.
- **Grades:** Find by grades.
- **Languages:** Locate vulnerable libraries by a specific language.
- **Usage:** Find by used or unused classes at runtime.
- **Licenses:** View libraries by licensed applications.
- **Environments:** Helps to easily locate any vulnerable libraries in production.
- **Servers:** Find vulnerable libraries by server type.
- **Library Severity:** Find by library severity.
- **Repositories:** Find by repository name.
- **Projects:** Find projects using the library.

More information about the filter types is available below. Note that some of the filters are visible under the static tab and not the runtime tab and vice versa.

Select **Show library stats** above the grid to analyze library data for your organization. Each graphic displays the statistical average as well as breakdowns for each category, including library scores and the number of years by which they are high risk. A library is considered high risk if it has a [score \(page 934\)](#) that is grade C or below.

Static and runtime tabs

Library information in Contrast is divided into two tabs:

- **Static:** Contains results from a [manifest](#) (for example, *package.json* or *pom.xml*) analyzed with [Contrast CLI \(page 1003\)](#).
- **Runtime:** Contains results for applications analyzed at runtime

Quick Views

Select a Quick View

Vulnerable (240)

Filters

Applications

Tags

Grades

Languages

Usage

Licenses

Environments

Servers

Library Severity

Libraries

Search

Static

Runtime

Close filters

<input type="checkbox"/>	Score	Library	Latest Version	Vulnerabilities (CVEs)	Applications	Usage	Actions
<input type="checkbox"/>	<div><div>F</div></div>	<div>spring-core-4.3.6.release.jar</div> <div>v4.3.6.RELEASE (Jan 25, 2017)</div> <div><div>custom</div><div>cve</div><div>remediate</div><div>tag1</div></div>	<div>v6.1.5</div> <div>Mar 14, 2024</div>	<div><div><div></div></div></div> <div>(8)</div>	<div>terraccotta</div> <div>terraccotta-wdd</div>	<div>462/789</div>	<div><div>info</div><div>refresh</div><div>delete</div></div>
<input type="checkbox"/>	<div><div>F</div></div>	<div>snakeyaml-1.17.jar</div> <div>v1.17 (Feb 19, 2016)</div> <div><div>remediate</div><div>tagtest1</div></div>	<div>v2.2</div> <div>Aug 27, 2023</div>	<div><div><div></div></div></div> <div>(8)</div>	<div>petclinic-wdd</div> <div>terraccotta</div> <div>terraccotta-wdd</div>	<div>Unused</div>	<div><div>info</div><div>refresh</div><div>delete</div></div>
<input type="checkbox"/>	<div><div>F</div></div>	<div>logback-classic-1.2.3.jar</div> <div>v1.2.3 (Mar 21, 2017)</div> <div><div>custom</div><div>cve</div><div>firstrun</div><div>remediate</div><div>sd</div></div>	<div>v1.5.3</div> <div>Mar 4, 2024</div>	<div><div><div></div></div></div> <div>(1)</div>	<div>ts-24115-stg</div> <div>terraccotta</div> <div>terraccotta-wdd</div>	<div>71/175</div>	<div><div>info</div><div>refresh</div><div>delete</div></div>
<input type="checkbox"/>	<div><div>F</div></div>	<div>spring-boot-starter-web-1.5.1.release.jar</div> <div>v1.5.1.RELEASE (Jan 30, 2017)</div>	<div>v3.2.3</div> <div>Feb 22, 2024</div>	<div><div><div></div></div></div> <div>(1)</div>	<div>terraccotta</div> <div>terraccotta-wdd</div>	<div>Unused</div>	<div><div>info</div><div>refresh</div><div>delete</div></div>
<input type="checkbox"/>	<div><div>F</div></div>	<div>logback-core-1.2.3.jar</div> <div>v1.2.3 (Mar 21, 2017)</div>	<div>v1.5.3</div> <div>Mar 4, 2024</div>	<div><div><div></div></div></div> <div>(2)</div>	<div>ts-24115-stg</div> <div>terraccotta</div> <div>terraccotta-wdd</div>	<div>181/373</div>	<div><div>info</div><div>refresh</div><div>delete</div></div>
<input type="checkbox"/>	<div><div>F</div></div>	<div>spring-web-4.3.6.release.jar</div> <div>v4.3.6.RELEASE (Jan 25, 2017)</div>	<div>v6.1.5</div> <div>Mar 14, 2024</div>	<div><div><div></div></div></div> <div>(2)</div>	<div>terraccotta</div> <div>terraccotta-wdd</div>	<div>207/558</div>	<div><div>info</div><div>refresh</div><div>delete</div></div>
<input type="checkbox"/>	<div><div>F</div></div>	<div>spring-data-jpa-1.11.0.release.jar</div> <div>v1.11.0.RELEASE (Jan 26, 2017)</div> <div><div>cve</div><div>remediate</div><div>sdts</div></div>	<div>v3.2.3</div> <div>Feb 16, 2024</div>	<div><div><div></div></div></div> <div>(2)</div>	<div>terraccotta</div> <div>terraccotta-wdd</div>	<div>23/169</div>	<div><div>info</div><div>refresh</div><div>delete</div></div>
<input type="checkbox"/>	<div><div>F</div></div>	<div>jackson-databind-2.8.6.jar</div> <div>v2.8.6 (Jan 11, 2017)</div>	<div>v2.17.0</div> <div>Mar 12, 2024</div>	<div><div><div></div></div></div> <div>(54)</div>	<div>terraccotta</div> <div>terraccotta-wdd</div>	<div>264/582</div>	<div><div>info</div><div>refresh</div><div>delete</div></div>
<input type="checkbox"/>	<div><div>F</div></div>	<div>hsqldb-2.3.3.jar</div> <div>v2.3.3 (Jan 30, 2015)</div> <div><div>firstrun</div><div>remediate</div><div>tag1</div></div>	<div>v2.7.2</div> <div>May 30, 2023</div>	<div><div><div></div></div></div> <div>(1)</div>	<div>terraccotta</div> <div>terraccotta-wdd</div>	<div>264/587</div>	<div><div>info</div><div>refresh</div><div>delete</div></div>
<input type="checkbox"/>	<div><div>F</div></div>	<div>spring-webmvc-4.3.6.release.jar</div> <div>v4.3.6.RELEASE (Jan 25, 2017)</div>	<div>v6.1.5</div> <div>Mar 14, 2024</div>	<div><div><div></div></div></div> <div>(1)</div>	<div>terraccotta</div> <div>terraccotta-wdd</div>	<div>186/498</div>	<div><div>info</div><div>refresh</div><div>delete</div></div>

Page 3 of 24

Results per page

10

25

50

The libraries columns include:

- **Score:** Visible only under the Runtime tab. Shown as a letter grade using this [scoring guide \(page 934\)](#).
- **Severity:** Visible only under the Static tab. This represents the maximum severity level for all vulnerabilities (CVEs) present in the library. Use the filters to locate libraries based on severity level.

Note that the **Other** filter option locates any libraries with CVEs whose maximum severity is None (where CVSS score is 0) AND libraries without a CVE AND private or unknown libraries.

- **Library:** The name of the library.

Select a library name in the list to open the library details panel. The panel displays:

- A summary of the findings (*visible only under the Runtime tab*).
- Methods for fixing the detected vulnerabilities:
 - The minimum version of the library that has fewer vulnerabilities compared to the one you are using.
Use this version if upgrading to the least stable version is not practical or efficient in your environment.
 - The last stable version has the fewest vulnerabilities compared to the library you are using.
- A list of known vulnerabilities (CVEs) that Contrast found within the library along with a list of the applications and servers where the library appears.
- The EPSS (*Exploit Prediction Scoring System*) calculation which provides a probability range between 0 to 1 (0 and 100%). A higher score indicates a vulnerability likely will be exploited within 30 days.
- **Latest version:** Most recent library version.



NOTE

For .NET libraries. The **Latest version** value relates to the package upgrade recommendation. The library version and hash are determined by the file the Contrast agent detects. The hash represents the library file version while the upgrade version represents the package version.

- **Vulnerabilities (CVES):** This shows the CVEs found in the library and can help prioritize remediation. Hover over the thermometer section to see the number of CVEs by severity. Click the thermometer to open the details panel.



If vulnerabilities exist, they display as a list and are color-coded by severity. Vulnerabilities with a critical severity status appear at the top of the list and are coded red.

- **Applications:** *Visible only under the Runtime tab.* Lists applications using the library.
- **Usage:** *Visible only under the Runtime tab.* This shows the total number of classes used at runtime out of the total number of classes that are in the library. If none of the classes have been used at runtime, this column shows "Unused." When your application loads a class, the Contrast agent reports usage. If the class has not been used before, the usage decreases. Click the number to [analyze the library usage \(page 930\)](#). There you can see information on classes loaded as well as the risks and policy violations associated with the library.
- **Actions:** *Visible only under the Runtime tab.* This is where you can [tag \(page 929\)](#), [send \(page 930\)](#), or [delete \(page 928\)](#) the library.
- **Status:** *Visible only under the Runtime tab and requires a minimum of the [Edit \(page 1267\)](#) organization role to be able to change the status.* (Contact Support to request enabling this column if not visible for your organization). Visible under the **Applications > Application name > Libraries** tab. There are three types to view/apply:
 - **Not a problem:** This library has acknowledged vulnerabilities, and the risks are acceptable, or the library is unused.
 - **Remediated:** The vulnerable library has been remediated.
 - **Reported:** When a library with vulnerabilities is detected by Contrast.

- **Projects:** *Visible only under the Static tab.* Lists the projects using the library.

CVE details

To view the CVE details card, select the **Library** or **Vulnerabilites (CVES)** link and then select the CVE link under the *What's the risk?* section.

High

CVE-2023-6378

First seen in Contrast: 06/26/2024

NVD Published: 11/29/2023

NVD Last Modified: 11/29/2023

See NVD for latest information

See in cve.org

Severity and Metrics

CVSS v3.1

7.5

Impact Score

3.6

Exploitability Score

3.9

EPSS

0

Vector

CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H

Attack vector (AV)

Network

Attack complexity (AC)

Low

Privileges required (PR)

None

User interaction (UI)

None

Scope (S)

Unchanged

Confidentiality (C)

None

Integrity (I)

None

Availability (A)

High

Show less

Description

A serialization vulnerability in logback receiver component part of logback version 1.4.11 allows an attacker to mount a Denial-Of-Service attack by sending poisoned data.

Organizational Impact

Applications

28/7510 (0.37283623)%

Servers

28/34 (82.35294)%

Applications

Servers

Web-Application-wil

spring-petclinic-2.6.0

webgoat-adr-demo

spring-petclinic-2.6.0

Web-Application-bsow-adr-5

Web-Application-wil

Web-Application-thib

Splunk-Adr-Test

petclinic-protect

webgoat-adr-demo

petclinic-protect

Splunk-Adr-Test

Web-Application-thib

Web-Application-bsow-adr-5

- **First Seen in Contrast** provides insight into your exposure window
- **Organizational Impact** displays precisely which applications and servers are affected
- Display of **CVSS**, **Impact Score**, **Exploitability Score**, and **EPSS** severity and metrics add comprehensive risk assessment and help prioritization for remediation. The EPSS (*Exploit Prediction Scoring System*) calculation provides a probability range between 0 to 1 (0 and 100%). A higher score indicates a vulnerability likely to be exploited within 30 days.
- Select **See NVD for latest information** or **See in cve.org** to view information about the specific CVE on the site. Note that the NVD site only provides a snapshot of information when the CVE was raised and may not be the most current description of the CVE.

Discover or delete libraries



NOTE

Visible under Runtime view only. Not available for libraries statically analyzed.

Libraries are associated with the applications that use them and the servers where these applications are deployed.

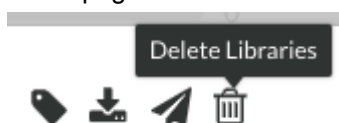
If an agent checks in with a library list that no longer includes a previously reported library, that library will no longer be associated with that server.

A library will be removed from an application once all servers reporting it have either checked in without it or the servers reporting it have been deleted. (Servers check in according to their [settings \(page 1171\)](#)).

Libraries can also be manually deleted from an application.

To delete libraries:

1. Select **Libraries** in the header and locate the row in the grid for the library you want to delete.
2. Click the **Delete** icon under the **Actions** column. You can also find the icon in the top right of the library details page.
To delete multiple libraries at once, use the check marks in the left column to select the libraries you want to delete, then select the **Delete** icon from the batch action bar that appears at the bottom of the page.



3. In the window that appears, select **Delete** to confirm your choice. Once confirmed, the library is removed and no longer appears in your list. If an agent reports a previously deleted library, this will be added to the list of libraries again as this library is included in an application.

Add tags to libraries

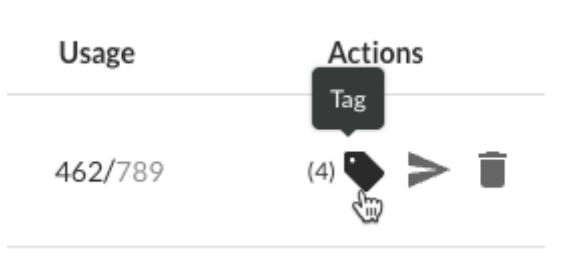


NOTE

Visible under Runtime view only. Not available for libraries statically analyzed.

To add tags to libraries:

1. Select **Libraries** in the header and locate the row in the grid for the library you want to tag.
2. Select the **Tag** icon under the **Actions** column. This option is also available from the library details page in the top right corner.



3. In the window that appears, begin typing to see a list of tags. Select one or more from the dropdown, and/or type a new tag. To remove tags, select the **X**. Select **Save**.
4. To tag multiple libraries, use the check marks in the left column of the libraries grid to select libraries. In the batch action menu that appears at the bottom of the page, select the **Tag** icon.
5. To filter by tags, select the open/close filter option on the left-side and then expand the Tags option, then select the tags to filter.

- You can also see tags next to the library name on the library's details page, and remove them by selecting the **X**.

Send library information



NOTE

Visible under Runtime view only. Not available for libraries statically analyzed.

Send library details to your email address or to an integrated bugtracker service (that creates tickets for your developers) to track vulnerable libraries.

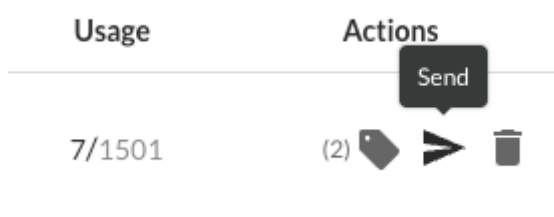
Contrast sends the following data to your email address or integrated bugtracker for each library that you choose.

- Name
- Version
- Vulnerabilities details
- Impacted applications and servers
- Versions behind (compared to the current/latest version)
- Usage (Currently available for Java and .NET only.)
- [Score \(page 934\)](#)

To receive this information for a particular library:

- Select **Libraries** in the header and locate the row in the grid for the library you want to track.
- Click the **Send** icon under the **Actions** column and select either **Send by Email** or **Send to Bugtracker**.

This option is also available from the library details page in the top right corner.



- In the window that appears, select the [integrations \(page 1051\)](#) you want to use. Select the options then select **Send** to create the ticket. For email, enter the email address then select **Send** to send the email.
- To receive information for multiple libraries, use the check marks in the left column of the libraries grid to select libraries. In the batch action menu that appears at the bottom of the page, select the **Send** icon. All libraries selected must have at least one application in common.

Analyze runtime library usage

Runtime library usage gives insight into which parts of a library are used by your applications, which can reduce investigation time for CVEs by showing how much a library impacts your application. This also improves collaboration because security teams can confirm with development teams that an application uses a vulnerable library at runtime.



NOTE

Only organizations with a [Contrast SCA license \(page 29\)](#) can see full usage details. To learn more, contact our sales department at sales@contrastsecurity.com.

Select **Libraries** in the header, click **Runtime** and view the **Usage** column to see if, and how much, a library is used at runtime. The usage number represents the number of items used by any instrumented applications, out of the total number of items known to be available in that library.

Libraries

Q

Search

Static

Runtime

Sort by

Score

Open filters

<input type="checkbox"/>	Score	Library	Latest Version	Vulnerabilities (CVES)	Applications	Usage	Actions
<input type="checkbox"/>	<div><div>F</div></div>	<div>spring-core-4.3.4.release.jar</div> <div>v4.3.6.RELEASE (Jan 25, 2017)</div> <div><div>custom</div><div>cve</div><div>remediate</div><div>tag1</div></div>	<div>v6.1.5</div> <div>Mar 14, 2024</div>	<div><div><div></div><div></div><div></div></div></div> <div>233</div> <div>(8)</div>	<div>terracotta</div> <div>terracotta-wdd</div>	<div>462/789</div> <div>(4)</div>	<div></div> <div></div> <div></div>
<input type="checkbox"/>	<div><div>F</div></div>	<div>snakeyaml-1.17.jar</div> <div>v1.17 (Feb 19, 2018)</div> <div><div>remediate</div><div>tagtest1</div></div>	<div>v2.2</div> <div>Aug 27, 2023</div>	<div><div><div></div><div></div><div></div></div></div> <div>125</div> <div>(8)</div>	<div>petclinic-wdd</div> <div>terracotta</div> <div>terracotta-wdd</div> <div>ts-24115-stg</div>	<div>1/175</div> <div>(6)</div>	<div></div> <div></div> <div></div>
<input type="checkbox"/>	<div><div>F</div></div>	<div>logback-classic-1.2.3.jar</div> <div>v1.2.3 (Mar 31, 2017)</div> <div><div>custom</div><div>cve</div><div>fixtrun</div><div>remediate</div><div>sdf</div></div>	<div>v1.5.3</div> <div>Mar 4, 2024</div>	<div><div><div></div></div></div> <div>1</div> <div>(1)</div>	<div>terracotta-wdd</div> <div>terracotta-wdd</div> <div>Show all (14)</div>	<div>1/175</div> <div>(6)</div>	<div></div> <div></div> <div></div>
<input type="checkbox"/>	<div><div>F</div></div>	<div>spring-boot-starter-web-1.5.1.release.jar</div> <div>v1.5.1.RELEASE (Jan 30, 2017)</div>	<div>v3.2.3</div> <div>Feb 22, 2024</div>	<div><div><div></div></div></div> <div>1</div> <div>(1)</div>	<div>terracotta-wdd</div> <div>terracotta-wdd</div> <div>ts-24115-stg</div>	<div>Unused</div>	<div></div> <div></div> <div></div>
<input type="checkbox"/>	<div><div>F</div></div>	<div>logback-core-1.2.3.jar</div>	<div>v1.5.3</div>	<div><div><div></div></div></div> <div></div> <div>(1)</div>	<div>terracotta-wdd</div> <div>ts-24115-stg</div> <div>terracotta</div>	<div>1/175</div>	<div></div> <div></div> <div></div>

Usage across all applications for this library

Max used / max known

Items loaded may be classes, files, or functions, depending on the languages of the applications using this library. If a primary application contains multiple applications that use the same library, the classes loaded will be representative of each merge application.

When an application uses a library, the Contrast agent reports the items loaded within the library. As the application uses more items within a library, usage counts increase in Contrast.

If you have the appropriate license, you can also view full library usage details for a particular application:

1. Under **Applications**, select a specific application to see the details view.
2. Select the **Libraries** tab for the application.
3. Select the usage counts for a specific library. This opens an overview and usage panel.

[illegible]

- Click the **Usage** tab to view each class, file, or function used. Click the search icon 🔍 to search for specific classes. You will also see the first time and last time Contrast observed it in use. [Library exports \(page 932\)](#) will also include full usage data. Click the **Overview** tab to view *What*

happened (describes the issue(s)), as well as *What's the risk* (lists the Severity badge, CVSS Score, CVE title, and policy violations).



NOTE

For a merged application, Contrast will report when a class was first seen and last seen for all applications it contains within the corresponding **Last Seen** and **First Seen** columns.

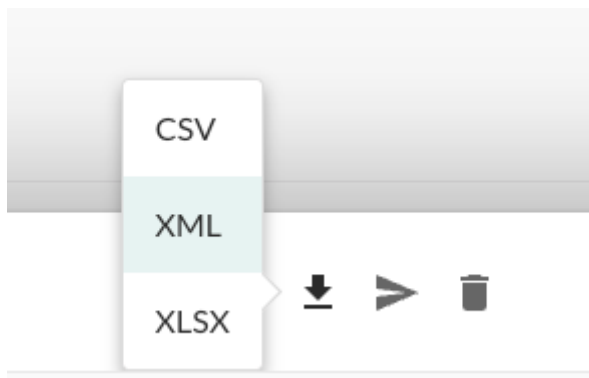
5. You can also [tag \(page 929\)](#) and [send \(page 930\)](#) information about the library.
6. Click the **More** menu (⋮) to view package details, repository location, [delete \(page 928\)](#) the library, or export usage details in CSV format.
7. Select the **X** to close the details panel.

Export library details

Use the export function to download library details.

To export library details:

1. Select **Libraries** in the header, then use the check marks in the left column of the libraries grid to select the library or libraries you want to use for the export. You can also click an individual library name to open the details panel.
2. In the batch action menu that appears at the bottom of the page, or in the upper right of the details panel, select the **Export** icon, then select the format you want to use for the export.



The export will download to your desktop. Note that the XLSX format export will contain two tabs: **Libraries** and **Vulnerabilities**.

Data fields

Exports contain the following data fields for each library:

- Library Name
- Language
- Version
- Release Date
- Latest Version
- Grade
- SHA1
- CVE Count
- Application Count
- Server Count
- Number of Classes

- Number of Used Classes
- Licenses
- App Name
- Server Name
- Server Environment
- Policy Violation
- Severity
- Tags

In addition to the above, the XLSX format export also contains a Vulnerabilities tab with the Hash, Library Name, CVE Name, Severity, and Severity Code fields.



TIP

To create more complex custom software composition analysis reports about your applications, you can use the [Libraries API](#) to access Contrast library data. You might also explore additional details on your libraries by using a manual method.

For example, this curl request retrieves a list of libraries in which each library includes a list of applications that use the library. The jq tool formats the data as CSV for use in a custom report.

```
$ curl -H "Authorization: $(echo -n $username:$servicekey  
base64)" -H "API-Key: $apikey" https://app.contrastsecurity.com/  
Contrast/api/ng/$org_id/libraries/filter?expand=apps  
jq -r '.libraries[]  
{name: .file_name, app_name: .apps[].name}  
[.name, .app_name]  
@csv'
```

View open-source licenses



IMPORTANT

Open-source license display is available to Contrast SCA customers only. Contact your Organization Administrator to enable SCA.

To view license details for your open-source libraries in SPDX format, select **Libraries** in the header. There are multiple ways to view license information:

- To view an individual library's license, hover over the library name in the grid.
- To find libraries with specific licenses, select the **filter** icon next to the **Library** column header and select the licenses you want to filter by.
- If you select a library name you will see details of that library. License information is also at the top of that page.
- Select **Package details** (either from the hover tip on the **Libraries** page, or by selecting the library name, then the information icon in the top right) to see the title, version and creator of that package.


- License details for each library are also included in any CSV or XML files [exported from Contrast \(page 932\)](#).

View dependency trees




When an open-source library is added to an application, all of the library's dependencies are also inherited. Some of these transitive dependencies may introduce vulnerable code into your applications. The [Contrast CLI \(page 994\)](#) identifies all library dependencies and sends the data to Contrast where you can visualize these libraries as a hierarchical dependency tree.

To display library hierarchy for your application, Contrast must have access to your application code at pre-compile time—a different stage of the software development lifecycle (SDLC) than the Contrast agents collect. To do this, you must have installed and run the [Contrast CLI \(page 994\)](#) for your applications.

To view an application's library dependency tree:

1. Select **Applications** in the navigation bar.
2. Select an application.
3. From the application's **Overview** page, select the **Libraries** tab.
4. Select the **dependency tree** icon  in the upper right to view the analysis of your application.

In this view, Contrast displays the dependency tree for your application's libraries based on the data collected by the [Contrast CLI \(page 994\)](#).

- Use the quick view menu to view only the vulnerabilities. By default, all libraries are displayed. You can use the right arrows to expand individual sections for more information or you can select the Expand All option to view all the information at once.
- Libraries with known vulnerabilities are also identified with a vulnerabilities warning icon . View vulnerability details by clicking the icon.
- Click the search icon  to search for a specific library.
- You can also view a dependency tree's history by choosing a custom date.
- Click the filter icon  to view the dependencies based on developer and/or production libraries. By default, the production option is selected.
- The **Application** dropdown appears for [merged \(page 613\)](#) applications so you can see how vulnerable libraries were introduced for the merged application. You can view the dependency tree by parent and child applications.

Library scoring guide

Contrast provides letter grades for the security of your application's libraries so that you can use them as a reference point during analysis. The grades map to scores as follows:

- A: 90 - 100
- B: 80 - 89
- C: 70 - 79
- D: 60 - 69
- F: 35 - 59

Scores are based on three penalty factors:

- **Time:** The age of the library is calculated based on the number of full years between the release of the latest version and the version used in the application, multiplied by 2.5.
- **Status:** The status is calculated based on the number of versions that have been released since the current library in your application, multiplied by 10.
- **Security:** The CVE penalty of the library is the highest severity of all known CVEs for this library, multiplied by 10.

**NOTE**

Organization administrators can [adjust the scoring method \(page 1177\)](#) to include only security criteria.

**TIP**

For example:

If you're using a library from January 2010 and the latest version came out in September 2013, the number of full years passed is two. So your time penalty would be:

$$2 \times 2.5 = 5$$

If you're using Version 1.1.1, but Versions 1.1.2 and 1.1.3 have been released, your penalty would be:

$$2 \times 10 = 20$$

If you have a library with the scores 2.4 and 2.2, the penalty would be:

$$2.4 \times 10 = 24$$

The final score of the library is calculated by subtracting each of the three penalty values from 100.

$$100 - 5 - 20 - 24 = 51$$

A score of 51 maps to a letter grade of F.

Libraries and Software Composition Analysis

Use Contrast's repository scanning capabilities to look for known vulnerabilities in the software components that are included in a repository. If a vulnerability is found, it will report the vulnerability to the repository owner. The owner can then take steps to fix the vulnerability or to mitigate the risk posed by the vulnerability.

Connect with repositories

Connect Contrast SCA to a GitHub, Bitbucket, or GitLab account and perform SCA scans.

- Connect with the [Contrast Security GitHub App \(page 936\)](#)
- Connect to [Bitbucket \(page 937\)](#)
- Connect to [GitLab \(page 937\)](#)

**NOTE**

Connections to Bitbucket and GitLab are available by request only. [Contact Support](#) to enable the connection.

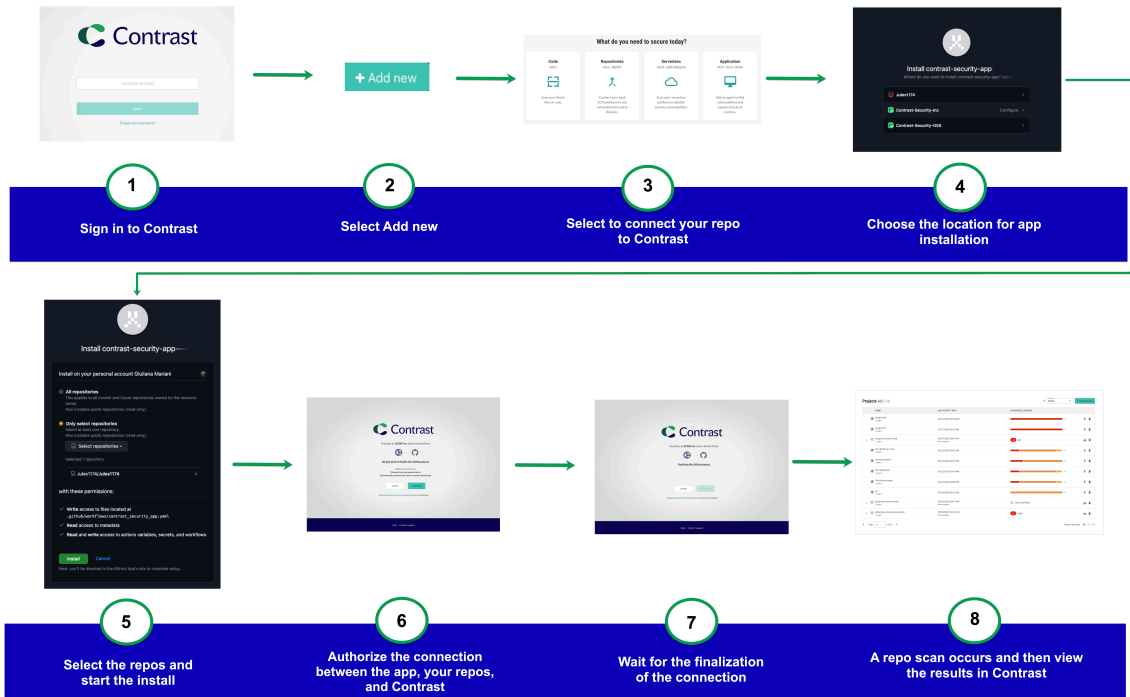
Contrast Security GitHub App

Use the Contrast Security GitHub App (also known as **Contrast Security SCA** in the GitHub Marketplace) to scan GitHub repositories with Contrast. Detect vulnerable libraries with how-to-fix guidance, and automate your CI/CD to prevent risk, at an earlier step, in your team's code.

How it works

For first-time use, sign in to Contrast, connect your GitHub account to Contrast and scan for library vulnerabilities in a repository.

Click the GitHub icon  to use the [Contrast Security GitHub App \(page 936\)](#) to connect with Contrast.



Once connected and scanned you can view the results in the [Projects \(page 600\)](#) list in Contrast.

You can also [connect from the GitHub Marketplace \(page 1042\)](#) with the Contrast Security GitHub App.

With this app, you can:

- Scan a GitHub repository
- Automate the security analysis of dependencies so that vulnerabilities can be detected and resolved during code review rather than after detection or exploitation in testing or production environments
- Any commits to the default branch and PRs created to merge into the default branch will trigger the workflow file. In addition, you can manually trigger the workflow.
- Users with edit, rules admin, or admin permissions will have access to the app

Next steps

- [Install and authorize \(page 1041\)](#)
- [GitHub repository connections \(page 1042\)](#)
- [Troubleshoot \(page 1043\)](#)


Bitbucket repositories with Contrast SCA



NOTE

Connection to Bitbucket is available by request only. [Contact Support](#) to enable the connection.

Connect to your Bitbucket repository to view scan results from Bitbucket files.

1. Click the Bitbucket icon  under the [Projects \(page 600\)](#) tab.
2. Select **Configure** to add projects to your workspace.
3. Select **Update** to establish the connection.




NOTE

A test commit is performed by Contrast before adding the changes to the bitbucket-pipelines.

Once connected, Contrast scans the repo and creates a group for each repo. You can find the results in the [Projects \(page 600\)](#) tab. If there are any updates to the repo, new scans will be triggered.

To disconnect from Bitbucket:

1. Click the Bitbucket icon  under the [Projects \(page 600\)](#) tab.
2. Select **Disconnect**.
3. Select the **Disconnect** button in the resulting window to disconnect all associated projects from Contrast.
To disconnect individual projects, locate the row in the grid for the project you want to disconnect and click the Delete icon.


GitLab repositories with Contrast SCA



NOTE


Connection to GitLab is available by request only. [Contact Support](#) to enable the connection.

Connect to your GitLab repository to view scan results from GitLab files. To connect to GitLab you must have a GitLab Owner or Maintainer role to write the required variables into a GitLab repository.

1. Click the GitLab icon  under the [Projects \(page 600\)](#) tab.
2. Select **Configure** to add projects to your namespace.
3. Select **Update** to establish the connection.

Once connected, Contrast scans the repo and creates a group for each repo. You can find the results in the [Projects \(page 600\)](#) tab. If there are any updates to the repo, new scans will be triggered.

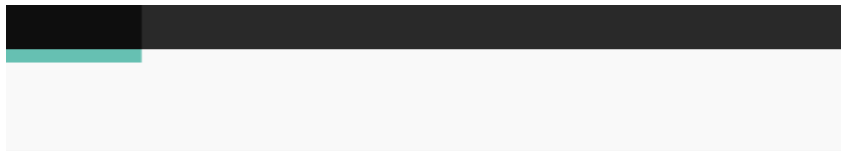
To disconnect from GitLab:

1. Click the GitLab icon  under the [Projects \(page 600\)](#) tab.
2. Select **Disconnect**.
3. Select the **Disconnect** button in the resulting window to disconnect all associated projects from Contrast.
To disconnect individual projects, locate the row in the grid for the project you want to disconnect and click the Delete icon.

CVE search

Use the search bar to locate specific CVEs in [libraries \(page 922\)](#).

- Enter a CVE number in the search bar and the search results will auto-populate with the libraries where the CVE is found.



- You can then select the relevant library and the library details will open.
- Search supported in [Organization \(page 1151\)](#), [Application \(page 602\)](#), and [Server \(page 911\)](#) libraries.

Contrast Serverless

Contrast Serverless features dynamic scanning, static scanning, graph visualization, and resource observability that help you maintain awareness of your environments.

With Contrast Serverless you can:

- [Scan functions on demand \(page 975\)](#)
- [View results \(page 977\)](#)
- [Change inventory criteria \(page 982\)](#)
- [Change Serverless scan settings \(page 983\)](#)
- [View function and service relationships \(page 984\)](#)

See also

- [Integrate Contrast Serverless with JIRA \(page 1100\)](#)

Contrast Serverless release notes

- [IDS release notes \(page 939\)](#)

IDS release notes

- [IDS Layer version 1.5.0 \(page 939\)](#)
- [IDS Layer version 1.4.0 \(page 939\)](#)
- [IDS Layer version 1.3.0 \(page 940\)](#)
- [IDS Layer version 1.2.0 \(page 941\)](#)
- [IDS Layer version 1.1.0 \(page 941\)](#)
- [IDS Layer version 1.0.0 \(page 942\)](#)

IDS Layer version 1.5.0

Release date: September 4, 2023

Language versions currently supported:

- Node.js 12, Node.js 14, Node.js 16
- Python 3.8, Python 3.9

Minimum requirements:

- Memory: 256 MB
- Timeout: 5 seconds

Included third-party packages: See [here \(page 942\)](#).

New and improved:

- Added a new ReDOS attack in Node.js
- Enhanced ReDOS instrumentation for Python to support more possible sinks
- Added a new `Unvalidated Input` attack in both Node.js and Python
- Added detection for Lambda triggers to enhance traces support
- Fixed false-positives in NOSQLI in DynamoDB
- We made several enhancements and addressed issues to enhance the overall performance and stability

Security fixes:

- Fixed CVE-2023-36665 `protobufjs` v7.1.1 in the Node.js layer
- Fixed CVE-2023-38704 `import-in-the-middle` v1.4.1 in the Node.js layer

Possible issues:

- Dependencies collision
 - See the included [third-party packages \(page 942\)](#)
- Node.js
 - Handler function that is defined with both `async` and a callback like `async function (event, context, callback)` is not supported
 - Webpack library target to `commonjs2`, with typescript compiler option `module` different than `Commonjs` is not supported

IDS Layer version 1.4.0

Release date: August 3, 2023

Language versions currently supported:

- Node.js 12, Node.js 14, Node.js 16

- Python 3.8, Python 3.9

Minimum requirements:

- Memory: 256 MB
- Timeout: 5 seconds

Included third-party packages: [See here \(page 942\).](#)

New and improved:

- Added a new ReDOS attack in Python (Node.js is coming in the next release)
- Added a new NoSQLI attack for DynamoDB and MongoDB in both Python and NodeJS
- Added support to improve validation of DAST attacks if the Lambda is instrumented with IDS
- Improved performance of the CMDI rule
- We made several enhancements and addressed issues to enhance overall performance and stability

Possible issues:

- Dependencies collision:
 - See the included [third-party packages \(page 942\).](#)
- Node.js:
 - Handler function that is defined with both `async` and a callback like `async function (event, context, callback)`, is not supported
 - Webpack library target to `commonjs2` with typescript compiler option module different than `Commonjs` is not supported

IDS Layer version 1.3.0

Release date: July 4, 2023

Language versions currently supported:

- Node.js 12, Node.js 14, Node.js 16
- Python 3.8, Python 3.9

Minimum requirements:

- Memory: 256 MB
- Timeout: 5 seconds

Included third-party packages: [See here \(page 942\).](#)

New and improved:

- Improved layer size (reduced by 10MB)
- Optimized the array input evaluation
- Added support for SQLI in DynamoDB
- Optimized array evaluation in nested keys
- Improved event type identification and handling

Possible issues:

- Dependencies collision:
 - See the included [third-party packages \(page 942\)](#)
- Node.js:
 - Handler function that is defined with both `async` and a callback like `async function (event, context, callback)`, is not supported

- Webpack library target to `commonjs2` with typescript compiler option module different than `Commonjs` is not supported

IDS Layer version 1.2.0

Release date: June 4, 2023

Language versions currently supported:

- Node.js 12, Node.js 14, Node.js 16
- Python 3.8, Python 3.9

Minimum requirements:

- Memory: 256 MB
- Timeout: 5 seconds

Included third-party packages: See [here \(page 942\)](#).

New and improved:

- Added support for events from service with "Records" like SQS, SNS, S3, etc.
- Optimized the rules evaluation during the attack phase
- Added initial support for traces
- Optimized the performance of Cold Starts

Bug fixes:

- Fixed unsupported syntax error in the NodeJS layer
- Fixed the evaluation of the rules - continue the evaluation even if one rule failed

Possible issues:

- Dependencies collision:
 - See the included third-party packages
- Node.js:
 - Handler function that is defined with both `async` and a callback like `async function (event, context, callback)`, is not supported
 - Webpack library target to `commonjs2` with typescript compiler option module different than `Commonjs` is not supported

IDS Layer version 1.1.0

Release date: May 22, 2023

Language versions currently supported:

- Node.js 12, Node.js 14, Node.js 16
- Python 3.8, Python 3.9

Minimum requirements:

- Memory: 256 MB
- Timeout: 5 seconds

Included third-party packages: See [here \(page 942\)](#).

New and improved:

- Removed false positive from LFI in Node.js
- Various performance optimizations

Security fixes:

- Prometheus vulnerability CVE-2019-3826

Possible issues:

- Dependencies collision:
 - See the included third-party packages
- Node.js
 - Handler function that is defined with both `async` and a callback like `async function (event, context, callback)`, is not supported
 - Webpack library target to `commonjs2` with typescript compiler option module different than `Commonjs` is not supported

IDS Layer version 1.0.0

Release date: May 11, 2023

Language versions currently supported:

- Node.js 12, Node.js 14, Node.js 16
- Python 3.8, Python 3.9

Minimum requirements:

- Memory: 256 MB
- Timeout: 5 seconds

Included third-party packages: See [here \(page 942\)](#).

New:

- The initial release of the IDS (Instrumented Dynamic Scan)
- Supported attacks:
 - OWASP Top 10 Serverless

Possible issues:

- Dependencies collision:
 - See the included third-party packages
- Node.js:
 - Handler function that is defined with both `async` and a callback like `async function (event, context, callback)`, is not supported
 - Webpack library target to `commonjs2` with typescript compiler option module different than `Commonjs` is not supported

Third-party packages

Layer Wrapper Dependencies

- [IDS Layer version 1.5.0 \(page 942\)](#)
- [IDS Layer version 1.4.0 \(page 948\)](#)
- [IDS Layer version 1.0.0 - 1.3.0 \(page 955\)](#)

IDS Layer version 1.5.0

The following are tables for the IDS Layer version 1.5.0.

Node.js

Name	Version
@cspotcode/source-map-support	0.8.1
@grpc/grpc-js	1.9.1
@grpc/proto-loader	0.7.9
@jridgewell/resolve-uri	3.1.1
@jridgewell/sourcemap-codec	1.4.15
@jridgewell/trace-mapping	0.3.9
@opentelemetry/api	1.4.1
@opentelemetry/api-logs	0.41.2
@opentelemetry/context-async-hooks	1.15.2
@opentelemetry/core	1.15.2
@opentelemetry/exporter-jaeger	1.15.2
@opentelemetry/exporter-metrics-otlp-http	0.41.2
@opentelemetry/exporter-metrics-otlp-proto	0.41.2
@opentelemetry/exporter-trace-otlp-grpc	0.41.2
@opentelemetry/exporter-trace-otlp-http	0.41.2
@opentelemetry/exporter-trace-otlp-proto	0.41.2
@opentelemetry/exporter-zipkin	1.15.2
@opentelemetry/instrumentation	0.41.2
@opentelemetry/instrumentation-aws-lambda	0.37.0
@opentelemetry/instrumentation-aws-sdk	0.36.0
@opentelemetry/instrumentation-fs	0.8.1
@opentelemetry/instrumentation-http	0.41.2
@opentelemetry/instrumentation-mongodb	0.37.0
@opentelemetry/instrumentation-mysql2	0.34.1
@opentelemetry/otlp-exporter-base	0.41.2
@opentelemetry/otlp-grpc-exporter-base	0.41.2
@opentelemetry/otlp-proto-exporter-base	0.41.2
@opentelemetry/otlp-transformer	0.41.2
@opentelemetry/propagation-utils	0.30.1
@opentelemetry/propagator-aws-xray	1.3.1
@opentelemetry/propagator-b3	1.15.2
@opentelemetry/propagator-jaeger	1.15.2
@opentelemetry/resource-detector-aws	1.3.1
@opentelemetry/resources	1.15.2
@opentelemetry/sdk-logs	0.41.2
@opentelemetry/sdk-metrics	1.15.2
@opentelemetry/sdk-node	0.41.2
@opentelemetry/sdk-trace-base	1.15.2
@opentelemetry/sdk-trace-node	1.15.2
@opentelemetry/semantic-conventions	1.15.2
@opentelemetry/sql-common	0.40.0
@protobufjs/aspromise	1.1.2
@protobufjs/base64	1.1.2
@protobufjs/codegen	2.0.4
@protobufjs/eventemitter	1.1.0
@protobufjs/fetch	1.1.0
@protobufjs/float	1.0.2
@protobufjs/inquire	1.1.0
@protobufjs/path	1.1.2

@protobufjs/pool	1.1.0
@protobufjs/utf8	1.1.0
@tsconfig/node10	1.0.9
@tsconfig/node12	1.0.11
@tsconfig/node14	1.0.3
@tsconfig/node16	1.0.4
@types/aws-lambda	8.10.119
@types/node	20.5.7
@types/shimmer	1.0.2
abbrev	1.1.1
accepts	1.3.8
acorn	8.10.0
acorn-import-assertions	1.9.0
acorn-walk	8.2.0
ansi-color	0.2.1
ansi-regex	5.0.1
ansi-styles	4.3.0
anymatch	3.1.3
arg	4.1.3
array-flatten	1.1.1
asynckit	0.4.0
available-typed-arrays	1.0.5
aws-sdk	2.1450.0
axios	1.5.0
balanced-match	1.0.2
base64-js	1.5.1
binary-extensions	2.2.0
body-parser	1.20.1
body-parser	1.20.2
brace-expansion	1.1.11
braces	3.0.2
buffer	4.9.2
bufw	1.3.0
bytes	3.1.2
call-bind	1.0.2
chokidar	3.5.3
cjs-module-lexer	1.2.3
cliui	8.0.1
cn-otel-node	0.0.1
color-convert	2.0.1
color-name	1.1.4
combined-stream	1.0.8
concat-map	0.0.1
content-disposition	0.5.4
content-type	1.0.5
cookie	0.5.0
cookie-signature	1.0.6
create-require	1.1.1
debug	2.6.9
debug	3.2.7
debug	4.3.4
delayed-stream	1.0.0

depd	2.0.0
destroy	1.2.0
diff	4.0.2
ee-first	1.1.1
emoji-regex	8.0.0
encodeurl	1.0.2
error	7.0.2
escalade	3.1.1
escape-html	1.0.3
etag	1.8.1
events	1.1.1
express	4.18.2
fill-range	7.0.1
finalhandler	1.2.0
follow-redirects	1.15.2
for-each	0.3.3
form-data	4.0.0
forwarded	0.2.0
fresh	0.5.2
fsevents	2.3.3
function-bind	1.1.1
get-caller-file	2.0.5
get-intrinsic	1.2.1
glob-parent	5.1.2
gopd	1.0.1
has	1.0.3
has-flag	3.0.0
has-proto	1.0.1
has-symbols	1.0.3
has-tostringtag	1.0.0
hexer	1.5.0
http-errors	2.0.0
iconv-lite	0.4.24
ieee754	1.1.13
ignore-by-default	1.0.1
import-in-the-middle	1.4.2
inherits	2.0.4
ipaddr.js	1.9.1
is-arguments	1.1.1
is-binary-path	2.1.0
is-callable	1.2.7
is-core-module	2.13.0
is-extglob	2.1.1
is-fullwidth-code-point	3.0.0
is-generator-function	1.0.10
is-glob	4.0.3
is-number	7.0.0
is-typed-array	1.1.12
isarray	1.0.0
jaeger-client	3.19.0
jmespath	0.16.0
lodash.camelcase	4.3.0

lodash.merge	4.6.2
long	2.4.0
long	5.2.3
lru-cache	6.0.0
make-error	1.3.6
media-typer	0.3.0
merge-descriptors	1.0.1
methods	1.1.2
mime	1.6.0
mime-db	1.52.0
mime-types	2.1.35
minimatch	3.1.2
minimist	1.2.8
module-details-from-path	1.0.3
ms	2.0.0
ms	2.1.2
ms	2.1.3
negotiator	0.6.3
node-int64	0.4.0
nodemon	3.0.1
nopt	1.0.10
normalize-path	3.0.0
object-inspect	1.12.3
on-finished	2.4.1
opentracing	0.14.7
parseurl	1.3.3
path-parse	1.0.7
path-to-regexp	0.1.7
picomatch	2.3.1
process	0.10.1
protobufjs	7.2.5
proxy-addr	2.0.7
proxy-from-env	1.1.0
pstree.remy	1.1.8
punycode	1.3.2
qs	6.11.0
querystring	0.2.0
range-parser	1.2.1
raw-body	2.5.1
raw-body	2.5.2
readdirp	3.6.0
require-directory	2.1.1
require-in-the-middle	7.2.0
resolve	1.22.4
safe-buffer	5.2.1
safer-buffer	2.1.2
sax	1.2.1
semver	7.5.4
send	0.18.0
serve-static	1.15.0
setprototypeof	1.2.0
shimmer	1.2.1

side-channel	1.0.4
simple-update-notifier	2.0.0
statuses	2.0.1
string-template	0.2.1
string-width	4.2.3
strip-ansi	6.0.1
supports-color	5.5.0
supports-preserve-symlinks-flag	1.0.0
thriftw	3.11.4
to-regexp-range	5.0.1
toidentifier	1.0.1
touch	3.1.0
ts-node	10.9.1
tsc	2.0.4
type-is	1.6.18
typescript	4.9.5
undefsafe	2.0.5
unpipe	1.0.0
url	0.10.3
util	0.12.5
utils-merge	1.0.1
uuid	8.0.0
uuid	8.3.2
v8-compile-cache-lib	3.0.1
vary	1.1.2
which-typed-array	1.1.11
wrap-ansi	7.0.0
xml2js	0.5.0
xmlbuilder	11.0.1
xorshift	1.2.0
xtend	4.0.2
y18n	5.0.8
yallist	4.0.0
yargs	17.7.2
yargs-parser	21.1.1
yn	3.1.1

Python

Name	Version
Deprecated	1.2.14
asgiref	3.7.2
backoff	2.2.1
certifi	2023.7.22
charset-normalizer	3.2.0
dnspython	2.4.2
googleapis-common-protos	1.60.0
idna	3.4
importlib-metadata	6.8.0
opentelemetry-api	1.19.0
opentelemetry-distro	0.40b0
opentelemetry-exporter-otlp-proto-common	1.19.0

opentelemetry-exporter-otlp-proto-http	1.19.0
opentelemetry-instrumentation	0.40b0
opentelemetry-instrumentation-aiohttp-client	0.40b0
opentelemetry-instrumentation-asgi	0.40b0
opentelemetry-instrumentation-asynccpg	0.40b0
opentelemetry-instrumentation-boto	0.40b0
opentelemetry-instrumentation-botocore	0.40b0
opentelemetry-instrumentation-celery	0.40b0
opentelemetry-instrumentation-dbapi	0.40b0
opentelemetry-instrumentation-django	0.40b0
opentelemetry-instrumentation-elasticsearch	0.40b0
opentelemetry-instrumentation-falcon	0.40b0
opentelemetry-instrumentation-fastapi	0.40b0
opentelemetry-instrumentation-flask	0.40b0
opentelemetry-instrumentation-grpc	0.40b0
opentelemetry-instrumentation-jinja2	0.40b0
opentelemetry-instrumentation-mysql	0.40b0
opentelemetry-instrumentation-psycpg2	0.40b0
opentelemetry-instrumentation-pymemcache	0.40b0
opentelemetry-instrumentation-pymongo	0.40b0
opentelemetry-instrumentation-pymysql	0.40b0
opentelemetry-instrumentation-pyramid	0.40b0
opentelemetry-instrumentation-redis	0.40b0
opentelemetry-instrumentation-requests	0.40b0
opentelemetry-instrumentation-sqlalchemy	0.40b0
opentelemetry-instrumentation-sqlite3	0.40b0
opentelemetry-instrumentation-starlette	0.40b0
opentelemetry-instrumentation-tornado	0.40b0
opentelemetry-instrumentation-urllib	0.40b0
opentelemetry-instrumentation-urllib3	0.40b0
opentelemetry-instrumentation-wsgi	0.40b0
opentelemetry-propagator-aws-xray	1.0.1
opentelemetry-proto	1.19.0
opentelemetry-sdk	1.19.0
opentelemetry-semantic-conventions	0.40b0
opentelemetry-util-http	0.40b0
packaging	23.1
protobuf	4.24.2
pymongo	4.5.0
requests	2.31.0
setuptools	68.1.2
typing_extensions	4.7.1
urllib3	2.0.4
wrapt	1.15.0
zipp	3.16.2

IDS Layer version 1.4.0

The following are tables for the IDS Layer version 1.4.0.

Node.js

Name	Version
------	---------

@cspotcode/source-map-support	0.8.1
@grpc/grpc-js	1.9.1
@grpc/proto-loader	0.7.9
@jridgewell/resolve-uri	3.1.1
@jridgewell/sourcemap-codec	1.4.15
@jridgewell/trace-mapping	0.3.9
@opentelemetry/api	1.4.1
@opentelemetry/api-logs	0.41.2
@opentelemetry/api-metrics	0.32.0
@opentelemetry/context-async-hooks	1.15.2
@opentelemetry/core	1.14.0
@opentelemetry/core	1.15.2
@opentelemetry/core	1.9.1
@opentelemetry/exporter-jaeger	1.15.2
@opentelemetry/exporter-metrics-otlp-http	0.41.2
@opentelemetry/exporter-metrics-otlp-proto	0.41.2
@opentelemetry/exporter-trace-otlp-grpc	0.35.1
@opentelemetry/exporter-trace-otlp-grpc	0.41.2
@opentelemetry/exporter-trace-otlp-http	0.35.1
@opentelemetry/exporter-trace-otlp-http	0.41.2
@opentelemetry/exporter-trace-otlp-proto	0.41.2
@opentelemetry/exporter-zipkin	1.15.2
@opentelemetry/instrumentation	0.32.0
@opentelemetry/instrumentation	0.35.1
@opentelemetry/instrumentation	0.40.0
@opentelemetry/instrumentation	0.41.2
@opentelemetry/instrumentation-aws-lambda	0.34.1
@opentelemetry/instrumentation-aws-sdk	0.36.0
@opentelemetry/instrumentation-fs	0.7.4
@opentelemetry/instrumentation-http	0.40.0
@opentelemetry/instrumentation-mongodb	0.36.1
@opentelemetry/instrumentation-mysql2	0.32.1
@opentelemetry/otlp-exporter-base	0.35.1
@opentelemetry/otlp-exporter-base	0.41.2
@opentelemetry/otlp-grpc-exporter-base	0.35.1
@opentelemetry/otlp-grpc-exporter-base	0.41.2
@opentelemetry/otlp-proto-exporter-base	0.41.2
@opentelemetry/otlp-transformer	0.35.1
@opentelemetry/otlp-transformer	0.41.2
@opentelemetry/propagation-utils	0.30.1
@opentelemetry/propagator-aws-xray	1.3.1
@opentelemetry/propagator-b3	1.15.2
@opentelemetry/propagator-jaeger	1.15.2
@opentelemetry/resource-detector-aws	1.3.1
@opentelemetry/resources	1.15.2
@opentelemetry/resources	1.9.1
@opentelemetry/sdk-logs	0.41.2
@opentelemetry/sdk-metrics	1.15.2
@opentelemetry/sdk-metrics	1.9.1
@opentelemetry/sdk-node	0.41.2
@opentelemetry/sdk-trace-base	1.15.2
@opentelemetry/sdk-trace-base	1.9.1

@opentelemetry/sdk-trace-node	1.15.2
@opentelemetry/semantic-conventions	1.14.0
@opentelemetry/semantic-conventions	1.15.2
@opentelemetry/semantic-conventions	1.9.1
@protobufjs/aspromise	1.1.2
@protobufjs/base64	1.1.2
@protobufjs/codegen	2.0.4
@protobufjs/eventemitter	1.1.0
@protobufjs/fetch	1.1.0
@protobufjs/float	1.0.2
@protobufjs/inquire	1.1.0
@protobufjs/path	1.1.2
@protobufjs/pool	1.1.0
@protobufjs/utf8	1.1.0
@tsconfig/node10	1.0.9
@tsconfig/node12	1.0.11
@tsconfig/node14	1.0.3
@tsconfig/node16	1.0.4
@types/aws-lambda	8.10.81
@types/node	20.5.7
@types/shimmer	1.0.2
abbrev	1.1.1
accepts	1.3.8
acorn	8.10.0
acorn-import-assertions	1.9.0
acorn-walk	8.2.0
ansi-color	0.2.1
ansi-regex	5.0.1
ansi-styles	4.3.0
anymatch	3.1.3
arg	4.1.3
array-flatten	1.1.1
asynckit	0.4.0
available-typed-arrays	1.0.5
aws-sdk	2.1448.0
axios	1.5.0
balanced-match	1.0.2
base64-js	1.5.1
binary-extensions	2.2.0
body-parser	1.20.1
body-parser	1.20.2
brace-expansion	1.1.11
braces	3.0.2
buffer	4.9.2
bufw	1.3.0
bytes	3.1.2
call-bind	1.0.2
chokidar	3.5.3
cjs-module-lexer	1.2.3
cliui	8.0.1
cn-otel-node	0.0.1
color-convert	2.0.1

color-name	1.1.4
combined-stream	1.0.8
concat-map	0.0.1
content-disposition	0.5.4
content-type	1.0.5
cookie	0.5.0
cookie-signature	1.0.6
create-require	1.1.1
debug	2.6.9
debug	3.2.7
debug	4.3.4
delayed-stream	1.0.0
depd	2.0.0
destroy	1.2.0
diff	4.0.2
ee-first	1.1.1
emoji-regex	8.0.0
encodeurl	1.0.2
error	7.0.2
escalade	3.1.1
escape-html	1.0.3
etag	1.8.1
events	1.1.1
express	4.18.2
fill-range	7.0.1
finalhandler	1.2.0
follow-redirects	1.15.2
for-each	0.3.3
form-data	4.0.0
forwarded	0.2.0
fresh	0.5.2
fsevents	2.3.3
function-bind	1.1.1
get-caller-file	2.0.5
get-intrinsic	1.2.1
glob-parent	5.1.2
gopd	1.0.1
has	1.0.3
has-flag	3.0.0
has-proto	1.0.1
has-symbols	1.0.3
has-tostringtag	1.0.0
hexer	1.5.0
http-errors	2.0.0
iconv-lite	0.4.24
ieee754	1.1.13
ignore-by-default	1.0.1
import-in-the-middle	1.3.5
import-in-the-middle	1.4.2
inherits	2.0.4
ipaddr.js	1.9.1
is-arguments	1.1.1

is-binary-path	2.1.0
is-callable	1.2.7
is-core-module	2.13.0
is-extglob	2.1.1
is-fullwidth-code-point	3.0.0
is-generator-function	1.0.10
is-glob	4.0.3
is-number	7.0.0
is-typed-array	1.1.12
isarray	1.0.0
jaeger-client	3.19.0
jmespath	0.16.0
lodash.camelcase	4.3.0
lodash.merge	4.6.2
long	2.4.0
long	5.2.3
lru-cache	6.0.0
make-error	1.3.6
media-typer	0.3.0
merge-descriptors	1.0.1
methods	1.1.2
mime	1.6.0
mime-db	1.52.0
mime-types	2.1.35
minimatch	3.1.2
minimist	1.2.8
module-details-from-path	1.0.3
ms	2.0.0
ms	2.1.2
ms	2.1.3
negotiator	0.6.3
node-int64	0.4.0
nodemon	3.0.1
nopt	1.0.10
normalize-path	3.0.0
object-inspect	1.12.3
on-finished	2.4.1
opentracing	0.14.7
parseurl	1.3.3
path-parse	1.0.7
path-to-regexp	0.1.7
picomatch	2.3.1
process	0.10.1
protobufjs	7.2.5
proxy-addr	2.0.7
proxy-from-env	1.1.0
pstree.remy	1.1.8
punycode	1.3.2
qs	6.11.0
querystring	0.2.0
range-parser	1.2.1
raw-body	2.5.1

raw-body	2.5.2
readdirp	3.6.0
require-directory	2.1.1
require-in-the-middle	5.2.0
require-in-the-middle	7.2.0
resolve	1.22.4
safe-buffer	5.2.1
safer-buffer	2.1.2
sax	1.2.1
semver	7.5.4
send	0.18.0
serve-static	1.15.0
setprototypeof	1.2.0
shimmer	1.2.1
side-channel	1.0.4
simple-update-notifier	2.0.0
statuses	2.0.1
string-template	0.2.1
string-width	4.2.3
strip-ansi	6.0.1
supports-color	5.5.0
supports-preserve-symlinks-flag	1.0.0
thriftw	3.11.4
to-regexp-range	5.0.1
toidentifier	1.0.1
touch	3.1.0
ts-node	10.9.1
tsc	2.0.4
type-is	1.6.18
typescript	4.9.5
undefsafe	2.0.5
unpipe	1.0.0
url	0.10.3
util	0.12.5
utils-merge	1.0.1
uuid	8.0.0
uuid	8.3.2
v8-compile-cache-lib	3.0.1
vary	1.1.2
which-typed-array	1.1.11
wrap-ansi	7.0.0
xml2js	0.5.0
xmlbuilder	11.0.1
xorshift	1.2.0
xtend	4.0.2
y18n	5.0.8
yallist	4.0.0
yargs	17.7.2
yargs-parser	21.1.1
yn	3.1.1

Python

Name	Version
Deprecated	1.2.14
asgiref	3.7.2
backoff	2.2.1
certifi	2023.7.22
charset-normalizer	3.2.0
dnspython	2.4.2
googleapis-common-protos	1.60.0
idna	3.4
importlib-metadata	6.0.0
importlib-metadata	6.8.0
opentelemetry-api	1.19.0
opentelemetry-distro	0.40b0
opentelemetry-exporter-otlp-proto-common	1.19.0
opentelemetry-exporter-otlp-proto-http	1.19.0
opentelemetry-instrumentation	0.40b0
opentelemetry-instrumentation-aiohttp-client	0.40b0
opentelemetry-instrumentation-asgi	0.40b0
opentelemetry-instrumentation-asynpg	0.40b0
opentelemetry-instrumentation-boto	0.40b0
opentelemetry-instrumentation-botocore	0.40b0
opentelemetry-instrumentation-celery	0.40b0
opentelemetry-instrumentation-dbapi	0.40b0
opentelemetry-instrumentation-django	0.40b0
opentelemetry-instrumentation-elasticsearch	0.40b0
opentelemetry-instrumentation-falcon	0.40b0
opentelemetry-instrumentation-fastapi	0.40b0
opentelemetry-instrumentation-flask	0.40b0
opentelemetry-instrumentation-grpc	0.40b0
opentelemetry-instrumentation-jinja2	0.40b0
opentelemetry-instrumentation-mysql	0.40b0
opentelemetry-instrumentation-psycopg2	0.40b0
opentelemetry-instrumentation-pymemcache	0.40b0
opentelemetry-instrumentation-pymongo	0.40b0
opentelemetry-instrumentation-pymysql	0.40b0
opentelemetry-instrumentation-pyramid	0.40b0
opentelemetry-instrumentation-redis	0.40b0
opentelemetry-instrumentation-requests	0.40b0
opentelemetry-instrumentation-sqlalchemy	0.40b0
opentelemetry-instrumentation-sqlite3	0.40b0
opentelemetry-instrumentation-starlette	0.40b0
opentelemetry-instrumentation-tornado	0.40b0
opentelemetry-instrumentation-urllib	0.40b0
opentelemetry-instrumentation-urllib3	0.40b0
opentelemetry-instrumentation-wsgi	0.40b0
opentelemetry-propagator-aws-xray	1.0.1
opentelemetry-proto	1.19.0
opentelemetry-sdk	1.19.0
opentelemetry-semantic-conventions	0.40b0
opentelemetry-util-http	0.40b0

packaging	23.1
protobuf	4.24.2
pymongo	4.5.0
requests	2.31.0
setuptools	68.1.2
typing_extensions	4.7.1
urllib3	2.0.4
wrapt	1.15.0
zipp	3.16.2

IDS Layer version 1.0.0 - 1.3.0

The following are tables for the IDS Layer versions 1.0.0 to 1.3.0.

Node.js

name	version
@babel/code-frame	7.12.11
@babel/helper-validator-identifier	7.19.1
@babel/highlight	7.18.6
@cspotcode/source-map-support	0.8.1
@eslint/eslintrc	0.4.3
@grpc/grpc-js	1.8.14
@grpc/proto-loader	0.7.7
@humanwhocodes/config-array	0.5.0
@humanwhocodes/object-schema	1.2.1
@isaacs/cliui	8.0.2
@jridgewell/resolve-uri	3.1.1
@jridgewell/sourcemap-codec	1.4.15
@jridgewell/trace-mapping	0.3.9
@nodelib/fs.scandir	2.1.5
@nodelib/fs.stat	2.0.5
@nodelib/fs.walk	1.2.8
@opentelemetry/api	1.3.0
@opentelemetry/api	1.4.1
@opentelemetry/api-metrics	0.32.0
@opentelemetry/context-async-hooks	1.13.0
@opentelemetry/core	1.13.0
@opentelemetry/exporter-jaeger	1.13.0
@opentelemetry/exporter-metrics-otlp-http	0.34.0
@opentelemetry/exporter-metrics-otlp-proto	0.34.0
@opentelemetry/exporter-trace-otlp-grpc	0.35.1
@opentelemetry/exporter-trace-otlp-http	0.35.1
@opentelemetry/exporter-trace-otlp-proto	0.34.0
@opentelemetry/exporter-zipkin	1.8.0
@opentelemetry/instrumentation	0.35.1
@opentelemetry/instrumentation	0.39.1
@opentelemetry/instrumentation-aws-lambda	0.34.1
@opentelemetry/instrumentation-aws-sdk	0.33.0
@opentelemetry/instrumentation-fs	0.7.2
@opentelemetry/instrumentation-mysql2	0.32.1
@opentelemetry/otlp-exporter-base	0.34.0

@opentelemetry/otlp-grpc-exporter-base	0.35.1
@opentelemetry/otlp-proto-exporter-base	0.34.0
@opentelemetry/otlp-transformer	0.34.0
@opentelemetry/propagation-utils	0.29.3
@opentelemetry/propagator-aws-xray	1.2.0
@opentelemetry/propagator-b3	1.13.0
@opentelemetry/propagator-jaeger	1.13.0
@opentelemetry/resource-detector-aws	1.2.3
@opentelemetry/resources	1.13.0
@opentelemetry/sdk-metrics	1.8.0
@opentelemetry/sdk-node	0.34.0
@opentelemetry/sdk-trace-base	1.13.0
@opentelemetry/sdk-trace-node	1.13.0
@opentelemetry/semantic-conventions	1.13.0
@pkgjs/parseargs	0.11.0
@protobufjs/aspromise	1.1.2
@protobufjs/base64	1.1.2
@protobufjs/codegen	2.0.4
@protobufjs/eventemitter	1.1.0
@protobufjs/fetch	1.1.0
@protobufjs/float	1.0.2
@protobufjs/inquire	1.1.0
@protobufjs/path	1.1.2
@protobufjs/pool	1.1.0
@protobufjs/utf8	1.1.0
@tsconfig/node10	1.0.9
@tsconfig/node12	1.0.11
@tsconfig/node14	1.0.3
@tsconfig/node16	1.0.4
@types/aws-lambda	8.10.81
@types/json-schema	7.0.11
@types/long	4.0.2
@types/minimist	1.2.2
@types/mocha	10.0.1
@types/mysql	git+ssh://git@github.com/types/ mysql.git#c26b1bc2bac17010081455e3127a90fb2eafcec9
@types/mysql2	git+ssh://git@github.com/types/ mysql2.git#89378b2cb3974ea8cdd1d633b8f056e54e5d2384
@types/node	18.16.9
@types/node	20.1.4
@types/normalize-package-data	2.4.1
@types/semver	7.5.0
@typescript-eslint/eslint-plugin	4.33.0
@typescript-eslint/experimental-utils	4.33.0
@typescript-eslint/parser	4.33.0
@typescript-eslint/scope-manager	4.33.0
@typescript-eslint/types	4.33.0
@typescript-eslint/typescript-estree	4.33.0
@typescript-eslint/visitor-keys	4.33.0
abbrev	1.1.1
accepts	1.3.8
acorn	7.4.1

acorn	8.8.2
acorn-jsx	5.3.2
acorn-walk	8.2.0
ajv	6.12.6
ansi-color	0.2.1
ansi-colors	4.1.3
ansi-escapes	4.3.2
ansi-regex	5.0.1
ansi-styles	4.3.0
anymatch	3.1.3
arg	4.1.3
argparse	1.0.10
array-flatten	1.1.1
array-union	2.1.0
arrify	1.0.1
astral-regex	2.0.0
asyncit	0.4.0
available-typed-arrays	1.0.5
aws-sdk	2.1378.0
axios	1.4.0
balanced-match	1.0.2
base64-js	1.5.1
binary-extensions	2.2.0
body-parser	1.20.2
brace-expansion	1.1.11
braces	3.0.2
buffer	4.9.2
bufw	1.3.0
bytes	3.1.2
call-bind	1.0.2
callsites	3.1.0
camelcase	5.3.1
camelcase-keys	6.2.2
chalk	4.1.2
chardet	0.7.0
chokidar	3.5.3
cli-cursor	3.1.0
cli-width	3.0.0
cliui	8.0.1
color-convert	2.0.1
color-name	1.1.4
combined-stream	1.0.8
concat-map	0.0.1
content-disposition	0.5.4
content-type	1.0.5
cookie	0.5.0
cookie-signature	1.0.6
create-require	1.1.1
cross-spawn	7.0.3
debug	2.6.9
debug	4.3.4
decamelize	1.2.0

decamelize-keys	1.1.1
deep-is	0.1.4
delayed-stream	1.0.0
denque	1.5.1
depd	2.0.0
destroy	1.2.0
diff	4.0.2
dir-glob	3.0.1
doctrine	3.0.0
eastasianwidth	0.2.0
ee-first	1.1.1
emoji-regex	8.0.0
encodeurl	1.0.2
enquirer	2.3.6
error	7.0.2
error-ex	1.3.2
escalade	3.1.1
escape-html	1.0.3
escape-string-regexp	4.0.0
eslint	7.32.0
eslint-config-prettier	7.2.0
eslint-plugin-es	3.0.1
eslint-plugin-node	11.1.0
eslint-plugin-prettier	3.4.1
eslint-scope	5.1.1
eslint-utils	3.0.0
eslint-visitor-keys	2.1.0
espre	7.3.1
esprima	4.0.1
esquery	1.5.0
esrecurse	4.3.0
estraverse	4.3.0
esutils	2.0.3
etag	1.8.1
events	1.1.1
execa	5.1.1
express	4.18.2
external-editor	3.1.0
fast-deep-equal	3.1.3
fast-diff	1.2.0
fast-glob	3.2.12
fast-json-stable-stringify	2.1.0
fast-levenshtein	2.0.6
fastq	1.15.0
figures	3.2.0
file-entry-cache	6.0.1
fill-range	7.0.1
finalhandler	1.2.0
find-up	4.1.0
flat-cache	3.0.4
flatted	3.2.7
follow-redirects	1.15.2

for-each	0.3.3
foreground-child	3.1.1
form-data	4.0.0
forwarded	0.2.0
fresh	0.5.2
fs.realpath	1.0.0
fsevents	2.3.2
function-bind	1.1.1
functional-red-black-tree	1.0.1
generate-function	2.3.1
get-caller-file	2.0.5
get-intrinsic	1.2.1
get-stream	6.0.1
glob	10.2.4
glob-parent	5.1.2
globals	13.20.0
globby	11.1.0
gopd	1.0.1
gts	3.1.1
hard-rejection	2.1.0
has	1.0.3
has-flag	3.0.0
has-flag	4.0.0
has-proto	1.0.1
has-symbols	1.0.3
has-tostringtag	1.0.0
hexer	1.5.0
hosted-git-info	4.1.0
http-errors	2.0.0
human-signals	2.1.0
iconv-lite	0.4.24
ieee754	1.1.13
ignore	5.2.4
ignore-by-default	1.0.1
import-fresh	3.3.0
imurmurhash	0.1.4
indent-string	4.0.0
inflight	1.0.6
inherits	2.0.4
inquirer	7.3.3
ipaddr.js	1.9.1
is-arguments	1.1.1
is-arrayish	0.2.1
is-binary-path	2.1.0
is-callable	1.2.7
is-core-module	2.12.0
is-extglob	2.1.1
is-fullwidth-code-point	3.0.0
is-generator-function	1.0.10
is-glob	4.0.3
is-number	7.0.0
is-plain-obj	1.1.0

is-property	1.0.2
is-stream	2.0.1
is-typed-array	1.1.10
is-typedarray	1.0.0
isarray	1.0.0
isexe	2.0.0
jackspeak	2.2.0
jaeger-client	3.19.0
jmespath	0.16.0
js-tokens	4.0.0
js-yaml	3.14.1
json-parse-even-better-errors	2.3.1
json-schema-traverse	0.4.1
json-stable-stringify-without-jsonify	1.0.1
json5	2.2.3
kind-of	6.0.3
levn	0.4.1
lines-and-columns	1.2.4
locate-path	5.0.0
lodash	4.17.21
lodash.camelcase	4.3.0
lodash.merge	4.6.2
lodash.truncate	4.4.2
long	4.0.0
lru-cache	6.0.0
make-error	1.3.6
map-obj	4.3.0
media-typer	0.3.0
meow	9.0.0
merge-descriptors	1.0.1
merge-stream	2.0.0
merge2	1.4.1
methods	1.1.2
micromatch	4.0.5
mime	1.6.0
mime-db	1.52.0
mime-types	2.1.35
mimic-fn	2.1.0
min-indent	1.0.1
minimatch	3.1.2
minimist	1.2.8
minimist-options	4.1.0
minipass	6.0.0
module-details-from-path	1.0.3
ms	2.0.0
ms	2.1.2
mute-stream	0.0.8
mysql2	2.3.0
named-placeholders	1.1.3
natural-compare	1.4.0
ncp	2.0.0
negotiator	0.6.3

node-int64	0.4.0
nodemon	2.0.22
nopt	1.0.10
normalize-package-data	3.0.3
normalize-path	3.0.0
npm-run-path	4.0.1
object-inspect	1.12.3
on-finished	2.4.1
once	1.4.0
onetime	5.1.2
opentracing	0.14.7
optionator	0.9.1
os-tmpdir	1.0.2
p-limit	2.3.0
p-locate	4.1.0
p-try	2.2.0
parent-module	1.0.1
parse-json	5.2.0
parseurl	1.3.3
path-exists	4.0.0
path-is-absolute	1.0.1
path-key	3.1.1
path-parse	1.0.7
path-scurry	1.9.1
path-to-regexp	0.1.7
path-type	4.0.0
picomatch	2.3.1
prelude-ls	1.2.1
prettier	2.8.8
prettier-linter-helpers	1.0.0
process	0.10.1
progress	2.0.3
protobufjs	7.2.3
proxy-addr	2.0.7
proxy-from-env	1.1.0
pstree.remy	1.1.8
punycode	1.3.2
punycode	2.3.0
qs	6.11.0
querystring	0.2.0
queue-microtask	1.2.3
quick-lru	4.0.1
range-parser	1.2.1
raw-body	2.5.2
read-pkg	5.2.0
read-pkg-up	7.0.1
readdirp	3.6.0
redent	3.0.0
regexpp	3.2.0
require-directory	2.1.1
require-from-string	2.0.2
require-in-the-middle	5.2.0

require-in-the-middle	7.1.0
resolve	1.22.2
resolve-from	4.0.0
restore-cursor	3.1.0
reusify	1.0.4
rimraf	5.0.0
run-async	2.4.1
run-parallel	1.2.0
rxjs	6.6.7
safe-buffer	5.2.1
safer-buffer	2.1.2
sax	1.2.1
semver	7.5.1
send	0.18.0
seq-queue	0.0.5
serve-static	1.15.0
setprototypeof	1.2.0
shebang-command	2.0.0
shebang-regex	3.0.0
shimmer	1.2.1
side-channel	1.0.4
signal-exit	3.0.7
simple-update-notifier	1.1.0
slash	3.0.0
slice-ansi	4.0.0
spdx-correct	3.2.0
spdx-exceptions	2.3.0
spdx-expression-parse	3.0.1
spdx-license-ids	3.0.13
sprintf-js	1.0.3
sqlstring	2.3.3
statuses	2.0.1
string-template	0.2.1
string-width	4.2.3
string-width-cjs	npm:string-width@4.2.3
strip-ansi	6.0.1
strip-ansi-cjs	npm:strip-ansi@6.0.1
strip-final-newline	2.0.0
strip-indent	3.0.0
strip-json-comments	3.1.1
supports-color	5.5.0
supports-color	7.2.0
supports-preserve-symlinks-flag	1.0.0
table	6.8.1
text-table	0.2.0
thriftw	3.12.0
through	2.3.8
tmp	0.0.33
to-regex-range	5.0.1
toidentifier	1.0.1
touch	3.1.0
trim-newlines	3.0.1

ts-node	10.9.1
tsc	2.0.4
tslib	1.14.1
tsutils	3.21.0
type-check	0.4.0
type-fest	0.20.2
type-is	1.6.18
typedarray-to-buffer	3.1.5
typescript	4.9.5
typescript	5.0.4
undefsafe	2.0.5
unpipe	1.0.0
uri-js	4.4.1
url	0.10.3
util	0.12.5
utils-merge	1.0.1
uuid	8.0.0
v8-compile-cache	2.3.0
v8-compile-cache-lib	3.0.1
validate-npm-package-license	3.0.4
vary	1.1.2
which	2.0.2
which-typed-array	1.1.9
word-wrap	1.2.3
wrap-ansi	7.0.0
wrap-ansi	8.1.0
wrap-ansi-cjs	npm:wrap-ansi@7.0.0
wrappy	1.0.2
write-file-atomic	3.0.3
xml2js	0.5.0
xmlbuilder	11.0.1
xorshift	1.2.0
xtend	4.0.2
y18n	5.0.8
yallist	4.0.0
yargs	17.7.2
yargs-parser	20.2.9
yargs-parser	21.1.1
yn	3.1.1

Python

name	version
deprecated	1.2.13
asgiref	3.6.0
backoff	2.2.1
certifi	2023.5.7
charset-normalizer	3.1.0
googleapis-common-protos	1.59.0
idna	3.4
importlib-metadata	6.0.1
opentelemetry-api	1.17.0

opentelemetry-distro	0.38b.0
opentelemetry-distro	0.38b0
opentelemetry-exporter-otlp-proto-http	1.17.0
opentelemetry-instrumentation	0.38b0
opentelemetry-instrumentation-aiohttp-client	0.38b0
opentelemetry-instrumentation-asgi	0.38b0
opentelemetry-instrumentation-asynpg	0.38b0
opentelemetry-instrumentation-boto	0.38b0
opentelemetry-instrumentation-botocore	0.38b0
opentelemetry-instrumentation-celery	0.38b0
opentelemetry-instrumentation-dbapi	0.38b0
opentelemetry-instrumentation-django	0.38b0
opentelemetry-instrumentation-elasticsearch	0.38b0
opentelemetry-instrumentation-falcon	0.38b0
opentelemetry-instrumentation-fastapi	0.38b0
opentelemetry-instrumentation-flask	0.38b0
opentelemetry-instrumentation-grpc	0.38b0
opentelemetry-instrumentation-jinja2	0.38b0
opentelemetry-instrumentation-mysql	0.38b0
opentelemetry-instrumentation-psycpg2	0.38b0
opentelemetry-instrumentation-pymemcache	0.38b0
opentelemetry-instrumentation-pymongo	0.38b0
opentelemetry-instrumentation-pymysql	0.38b0
opentelemetry-instrumentation-pyramid	0.38b0
opentelemetry-instrumentation-redis	0.38b0
opentelemetry-instrumentation-requests	0.38b0
opentelemetry-instrumentation-sqlalchemy	0.38b0
opentelemetry-instrumentation-sqlite3	0.38b0
opentelemetry-instrumentation-starlette	0.38b0
opentelemetry-instrumentation-tornado	0.38b0
opentelemetry-instrumentation-wsgi	0.38b0
opentelemetry-propagator-aws-xray	1.0.1
opentelemetry-proto	1.17.0
opentelemetry-sdk	1.17.0
opentelemetry-semantic-conventions	0.38b0
opentelemetry-util-http	0.38b0
protobuf	4.23.0
requests	2.30.0
setuptools	67.7.2
typing_extensions	4.5.0
urllib3	2.0.2
wrapt	1.15.0
zipp	3.15.0

Contrast Serverless supported languages

The following programming languages are supported for Contrast Serverless.

Language	Runtime version
Java	8.x, 11.x, 17.x
.NET	.NET 5.x, 6.x .NET Core 3.1

Language	Runtime version
Node.js	12.x, 14.x, 16.x, 18.x
Python	3.6, 3.7, 3.8, 3.9

Contrast Serverless supported platforms

The following platforms are supported for Contrast Serverless.

- AWS Lambda
- Microsoft Azure

Multi-region support

Contrast Serverless provides support for multiple regions in a single AWS account to help organizations with teams that work within different regions.

With this, you will be able to onboard agents to multiple regions within the same AWS account. You will also be able to view and manage the account and region separately for each combination of account and region.



NOTE

All supported regions can be added to a single AWS account.

Supported regions

Contrast Serverless supports the following regions:

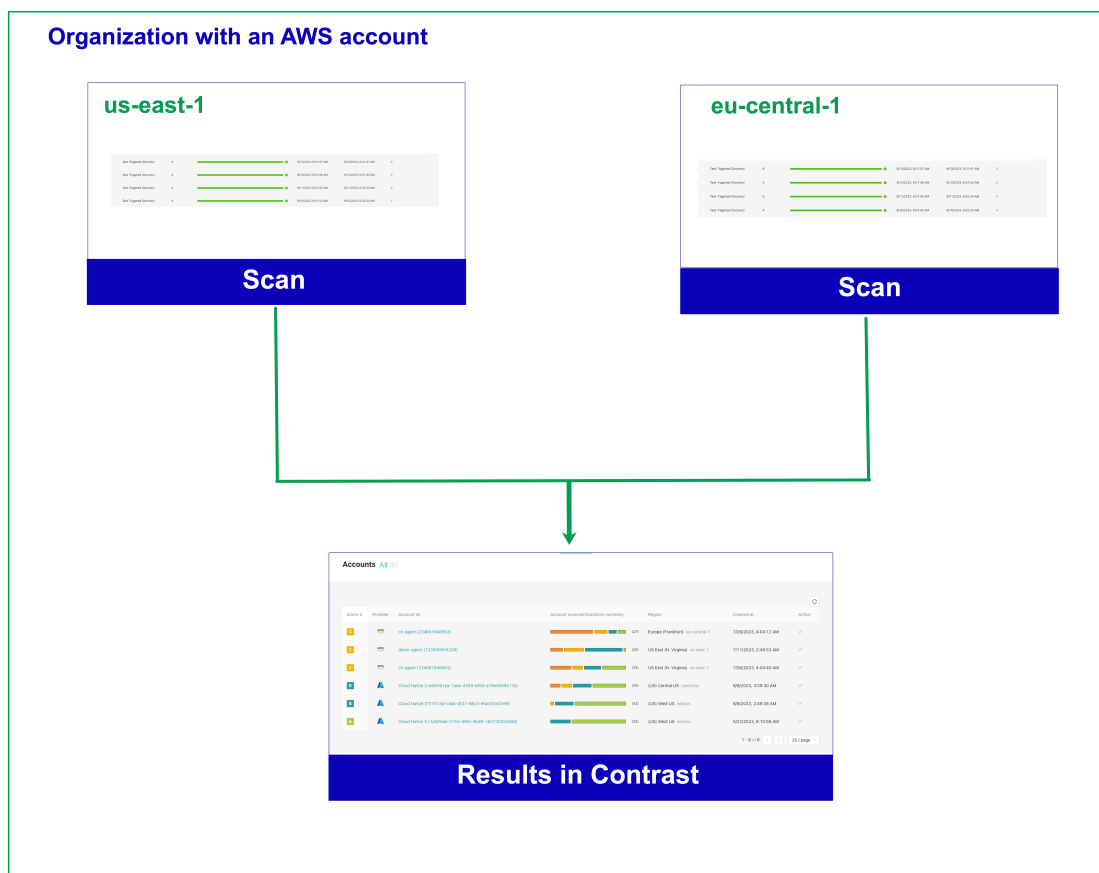
Serverless environment name	Site	Supported regions
AppUS	https://cs001.contrastsecurity.com https://app.contrastsecurity.com https://eval.contrastsecurity.com https://security-research.contrastsecurity.com https://ce.contrastsecurity.com	us-east-1 us-east-2 eu-west-1 eu-west-2 eu-central-1 eu-north-1 ap-northeast-1 ap-northeast-3 ap-southeast-1 us-west-1 us-west-2 ca-central-1
AppTWO	https://apptwo.contrastsecurity.com	us-east-1 eu-central-1 eu-west-1
AppJAPAN	https://app.contrastsecurity.jp	ap-northeast-1 ap-northeast-3
AppUK	https://cs002.contrastsecurity.com	eu-west-2
AppEU	https://eval003.contrastsecurity.com https://cs003.contrastsecurity.com	eu-west-1 eu-west-2 eu-central-1 us-east-1
Staging (dev1eu)	https://teamserver-staging.contsec.com	eu-central-1 us-east-1 eu-west-1

How it works

- With one AWS account for an organization, onboard the first region, then onboard the next region and so on.

Example: A development team has to onboard an account in *us-east-1*, which has the same applications (but different modules/code packages) used by another development team in *eu-central-1*.

- Once the regions are set up, you can run scans as usual and then see the results and graphs in Contrast.
- In Contrast, you will see the different on-boarded accounts based on region location.
- All account details including the region details, the number of functions in the account, the severity, and so on will be visible as usual in Contrast.
- You can also continue to manage the functions used by the team/region (based on tagging), and the ability to see all application functions on the graph, regardless of the different regions.



Inventory

When you connect to an AWS account from Contrast, it automatically discovers all Lambda functions and their relationships with different resources (such as S3, API Gateway, and DynamoDB) within the tested environment.

By default, Contrast scans all functions in the inventory unless [you specify otherwise. \(page 982\)](#)

Inventory criteria

Contrast lets you specify criteria that determine the scope of functions to be scanned, based on these criteria:

- **Tag:** This option lets you include or exclude functions associated with a specified tag and, optionally, a tag value.
- **Name:** This option lets you include or exclude functions with a specific name, prefix, or suffix.

Scan types and monitoring

Contrast Serverless supports these types of scans and monitoring:

Static scans

This scan automatically scans, in close to real-time, relevant static code and configuration assessments to discover new vulnerabilities in the following categories:

- **Least privilege:** Discovers IAM vulnerabilities (over permissive functions) within serverless workload before deployment and recommends permission remediations.
- **Contrast SCA** - Provides SCA for open-source libraries using the Contrast SCA engine.

The scan has no permanent effect on your code.

Dynamic scans

This scan type looks at dynamic assessments based on a specific update introduced to the tested environment. These scans are invoked with S3, API Gateway, and Dynamo DB functions.

It executes automatically, close to real-time, providing dynamic assessments, based on the specific update introduced to the tested environment. The dynamic scans are based on the interpretation of the OWASP Top 10 benchmark. For example:

- SQL injection
- Code injection
- Local file inclusion (LFI)

During a dynamic scan, Contrast tries to send malicious input to the code and then, exercises the code to discover vulnerabilities. This action does not affect your code, however, a scanned function is invoked.

Instrumented Dynamic analysis



NOTE

It is required to use *testing coverage* when performing an instrumented dynamic scan.

This option is recommended for selection when specifying scan settings.

In AWS accounts, instrumented dynamic analysis uncovers all exploitable AWS Lambda functions. With support for the latest AWS Lambda services you can uncover security issues in AWS Step Functions – a service that coordinates multiple Lambda functions into flexible workflows.

Uncover OWASP Top Ten vulnerabilities including:

- Injections (content, OS command, limited SQL, code)
- Cross-site scripting (XSS)
- Local file inclusion (LFI)

In addition, it provides improved AWS account observability and increased security coverage by uncovering all serverless account assets including *unused functions* (shadow functions). These

unused functions are usually not maintained and contain outdated dependencies leading to potential vulnerabilities.

Continuous monitoring

Once you connect to an account from Contrast, Contrast Serverless monitors this account. As you make changes to your functions' code or configurations, Contrast automatically initiates a new scan.

Get started with Contrast Serverless for AWS

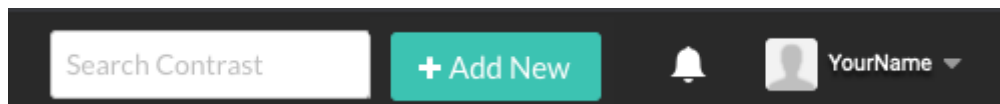
To start using Contrast Serverless, open Contrast and connect to your AWS account to create a new stack.

Before you begin

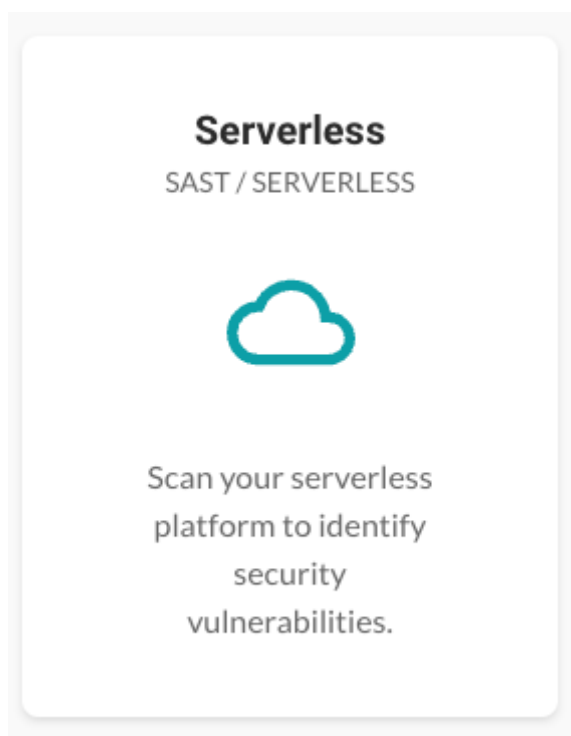
- Have your AWS account information available.
- Minimum [permissions \(page 969\)](#) required to deploy/update/delete a Contrast Serverless stack.

Steps

1. In the Contrast application, select **Add New** at the top of the page.



2. Select the **Serverless** card.



3. Select the **AWS** option under the **Cloud provider** section.
4. (Optional) Specify scan settings:
 - **Inventory:** Inventory consists of the functions that you want Contrast to scan. The default value is to scan all functions in your AWS account.
 - **Initial scan:** This setting determines actions that Contrast takes to scan your functions.

Static analysis	Dynamic analysis
<p>Covers:</p> <ul style="list-style-type: none"> • Least Privilege: Detects unused permissions. For Java, .NET Core 6, .NET Core 7, Node.js, and Python. • CVEs: Detects vulnerable dependencies. For Java, .NET Core 6, .NET Core 7, Node.js, and Python. • SAST: Detects custom-code vulnerabilities. For Java. • Malware: Detects malicious files. For Python. 	<p>Covers:</p> <ul style="list-style-type: none"> • The stress testing of an application to detect any possible vulnerabilities. • The Instrumented Dynamic analysis option enables Contrast Serverless to find function exploits in the entire account environment and across all services. See Scan types and monitoring (page 967) for more information. To get the analysis fully configured for your accounts, follow the steps under the <i>Instrumented Dynamic Scan Instructions</i> section. For Node.js and Python.

- **Deployment:** Deploy with a new stack in AWS or download the CFT to use in your pipeline. You can [change these settings \(page 983\)](#) at any time in the Settings tab.

5. Select **Create new stack**.
6. On the displayed AWS page, enter your account information and select **Create stack**. Alternatively, you can download a CloudFormation template and use it in your development pipeline. This action connects to the AWS CloudFormation Stacks console for your account and starts the first scan.
7. Approve the stack deployment in your account. The stack deployment takes approximately two minutes to complete.
8. Return to the Contrast application and verify that the Account is connected and the Scan started messages are displayed.
9. To view details about functions and scan results, select **function** in the Account connected message or select the **Serverless** tab.

Next steps

- [Scan functions on demand \(page 975\)](#)
- [View results \(page 977\)](#)
- [Change inventory criteria \(page 982\)](#)
- [Change scan settings \(page 983\)](#)

AWS policy and permissions for running Contrast Serverless

This is a sample of how to obtain the policy and permissions for your AWS account with Contrast Serverless.

Obtain a policy

This is a sample updated policy for an account.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CustomResources",
      "Effect": "Allow",
      "Action": [
        "sns:Publish"
      ],
      "Resource": "*"
    },
    {
      "Sid": "SNS2",
      "Effect": "Allow",
      "Action": [
```

```
        "sns:CreateTopic",
        "sns:GetTopicAttributes",
        "sns:DeleteTopic",
        "sns:TagResource"
    ],
    "Resource": "*"
},
{
    "Sid": "IAM",
    "Effect": "Allow",
    "Action": [
        "iam:AttachRolePolicy",
        "iam:CreatePolicy",
        "iam:CreateRole",
        "iam:CreateServiceLinkedRole",
        "iam:DeletePolicy",
        "iam:DeleteRole",
        "iam:DeleteRolePolicy",
        "iam:DetachRolePolicy",
        "iam:GetPolicy",
        "iam:GetRole",
        "iam:GetRolePolicy",
        "iam:ListPolicyVersions",
        "iam:ListRoleTags",
        "iam:PassRole",
        "iam:PutRolePolicy",
        "iam:TagRole",
        "iam:UntagRole"
    ],
    "Resource": "*"
},
{
    "Sid": "S3",
    "Effect": "Allow",
    "Action": [
        "s3:CreateBucket",
        "s3:DeleteBucket",
        "s3:DeleteBucketPolicy",
        "s3:GetBucketPolicy",
        "s3:PutBucketPolicy",
        "s3:PutBucketPublicAccessBlock",
        "s3:PutBucketTagging",
        "s3:PutEncryptionConfiguration",
        "s3:PutLifecycleConfiguration",
        "s3:PutBucketNotification"
    ],
    "Resource": "*"
},
{
    "Sid": "Lambda",
    "Effect": "Allow",
    "Action": [
        "lambda:GetFunction",
        "lambda:CreateFunction",
        "lambda:DeleteFunctionEventInvokeConfig",
```

```
        "lambda:DeleteFunction",
        "lambda:TagResource",
        "lambda:PutFunctionEventInvokeConfig"
    ],
    "Resource": "*"
},
{
    "Sid": "S3LambdaCode",
    "Effect": "Allow",
    "Action": [
        "s3:GetObject"
    ],
    "Resource": "*"
},
{
    "Sid": "EventsRule",
    "Effect": "Allow",
    "Action": [
        "events:DeleteRule",
        "events:DescribeRule",
        "events:PutRule",
        "events:PutTargets",
        "events:RemoveTargets"
    ],
    "Resource": "*"
},
{
    "Sid": "CloudTrail",
    "Effect": "Allow",
    "Action": [
        "cloudtrail:AddTags",
        "cloudtrail:CreateTrail",
        "cloudtrail>DeleteTrail",
        "cloudtrail:StartLogging",
        "cloudtrail:PutEventSelectors"
    ],
    "Resource": "*"
},
{
    "Sid": "CloudFormation",
    "Effect": "Allow",
    "Action": [
        "cloudformation:GetTemplateSummary",
        "cloudformation:CreateStack",
        "cloudformation:DescribeStackEvents"
    ],
    "Resource": "*"
}
]
```

Run the AWS iam create-policy:

```
```aws iam create-policy --policy-name Contrast-create-stack --policy-
document '{
 "Version": "2012-10-17",
```

```

"Statement": [
 {
 "Sid": "CustomResources",
 "Effect": "Allow",
 "Action": ["sns:Publish"],
 "Resource": "*"
 },
 {
 "Sid": "SNS2",
 "Effect": "Allow",
 "Action": [
 "sns:CreateTopic",
 "sns:GetTopicAttributes",
 "sns>DeleteTopic",
 "sns:TagResource"
],
 "Resource": "*"
 },
 {
 "Sid": "IAM",
 "Effect": "Allow",
 "Action": [
 "iam:AttachRolePolicy",
 "iam:CreatePolicy",
 "iam:CreateRole",
 "iam:CreateServiceLinkedRole",
 "iam>DeletePolicy",
 "iam>DeleteRole",
 "iam>DeleteRolePolicy",
 "iam:DetachRolePolicy",
 "iam:GetPolicy",
 "iam:GetRole",
 "iam:GetRolePolicy",
 "iam:ListPolicyVersions",
 "iam:ListRoleTags",
 "iam:PassRole",
 "iam:PutRolePolicy",
 "iam:TagRole",
 "iam:UntagRole"
],
 "Resource": "*"
 },
 {
 "Sid": "S3",
 "Effect": "Allow",
 "Action": [
 "s3:CreateBucket",
 "s3>DeleteBucket",
 "s3>DeleteBucketPolicy",
 "s3:GetBucketPolicy",
 "s3:PutBucketPolicy",
 "s3:PutBucketPublicAccessBlock",
 "s3:PutBucketTagging",
 "s3:PutEncryptionConfiguration",
 "s3:PutLifecycleConfiguration",

```



```
 "s3:PutBucketNotification"
],
 "Resource": "*"
},
{
 "Sid": "Lambda",
 "Effect": "Allow",
 "Action": [
 "lambda:GetFunction",
 "lambda:CreateFunction",
 "lambda:DeleteFunctionEventInvokeConfig",
 "lambda:DeleteFunction",
 "lambda:TagResource",
 "lambda:PutFunctionEventInvokeConfig"
],
 "Resource": "*"
},
{
 "Sid": "S3LambdaCode",
 "Effect": "Allow",
 "Action": ["s3:GetObject"],
 "Resource": "*"
},
{
 "Sid": "EventsRule",
 "Effect": "Allow",
 "Action": [
 "events:DeleteRule",
 "events:DescribeRule",
 "events:PutRule",
 "events:PutTargets",
 "events:RemoveTargets"
],
 "Resource": "*"
},
{
 "Sid": "CloudTrail",
 "Effect": "Allow",
 "Action": [
 "cloudtrail:AddTags",
 "cloudtrail:CreateTrail",
 "cloudtrail:DeleteTrail",
 "cloudtrail:StartLogging",
 "cloudtrail:PutEventSelectors"
],
 "Resource": "*"
},
{
 "Sid": "CloudFormation",
 "Effect": "Allow",
 "Action": [
 "cloudformation:GetTemplateSummary",
 "cloudformation:CreateStack",
 "cloudformation:DescribeStackEvents"
],
 "Resource": "*"
}
```

```
 "Resource": "*"
 }
}
}'''
```

And obtain a response:

```
{
 "Policy": { ...
 "PolicyName": "Contrast-serverless-create-stack",
 "PolicyId": "ANPAV3I66HSEE4ILG6XJ3",
 "Arn": "arn:aws:iam::402181209224:policy/Contrast-serverless-create-stack",
 "Path": "/",
 "DefaultVersionId": "v1",
 "AttachmentCount": 0,
 "PermissionsBoundaryUsageCount": 0,
 "IsAttachable": true,
 "CreateDate": "2023-01-24T16:36:49+00:00",
 "UpdateDate": "2023-01-24T16:36:49+00:00"
 }
}
```

Then attach a user policy:

```
aws iam attach-user-policy --policy-arn
arn:aws:iam::402181209224:policy/Contrast-serverless-create-stack --user-
name
<USER-NAME></USER-NAME>
```

And obtain a response:

```
4ppsec@TMEIAMED-C02DX3Q2ML85:~/cn-mono (*)
[> aws iam list-attached-user-policies --user-name xrays --profile xrays
{
 "AttachedPolicies": [
 {
 "PolicyName": "Contrast-serverless-create-stack",
 "PolicyArn": "arn:aws:iam::402181209224:policy/Contrast-serverless-create-stack"
 }
]
}
```

You can now run a deployment.

## Get started with Contrast Serverless for Azure

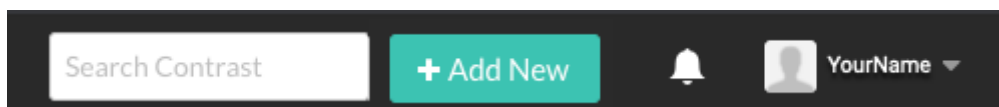
To start using Contrast Serverless with Azure, open Contrast and connect to your Azure account to create a new stack.

### Before you begin

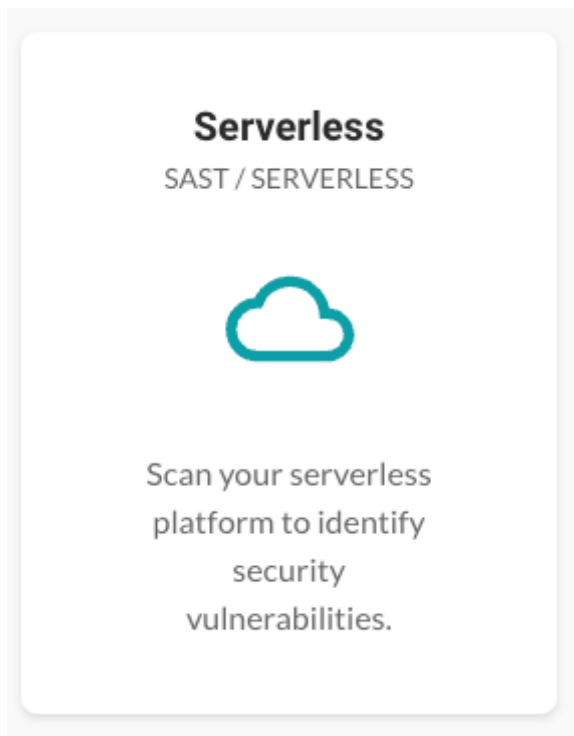
- Make sure you have an Azure account with an active subscription.
- Make sure you have permission to create an App registration on the Active Directory Tenant.
- Make sure you have an *Owner* role in the account. This allows for assigning roles to the App registrations.

### Steps

1. In the Contrast application, select **Add New** at the top of the page.



2. Select the **Serverless** card.



3. Select the **Azure** option under the **Cloud provider** section.
4. (Optional) Specify scan settings:
  - **Inventory:** Not available for Azure.
  - **Initial scan:** This setting determines actions that Contrast takes to scan your functions.

Static analysis	Dynamic analysis
<p>Covers:</p> <ul style="list-style-type: none"><li>• <b>Least Privilege</b>- Detects unused permissions. For Java, .NET Core 6, .NET Core 7, Node.js, and Python.</li><li>• <b>CVEs</b> - Detects vulnerable dependencies. For Java, .NET Core 6, .NET Core 7, Node.js, and Python.</li><li>• <b>SAST</b> - Detects custom-code vulnerabilities. For Java.</li><li>• <b>Malware</b> - Detects malicious files. For Python.</li></ul>	<p>Not available for Azure.</p>

You can [change these settings \(page 983\)](#) at any time in the Settings tab.

5. Continue with the steps under the **Deployment** section.
6. Return to the Contrast application and verify that the Account is connected and Scan started messages are displayed.

## Next steps

- [Scan functions on demand \(page 975\)](#)
- [View results \(page 977\)](#)

## Scan functions on demand

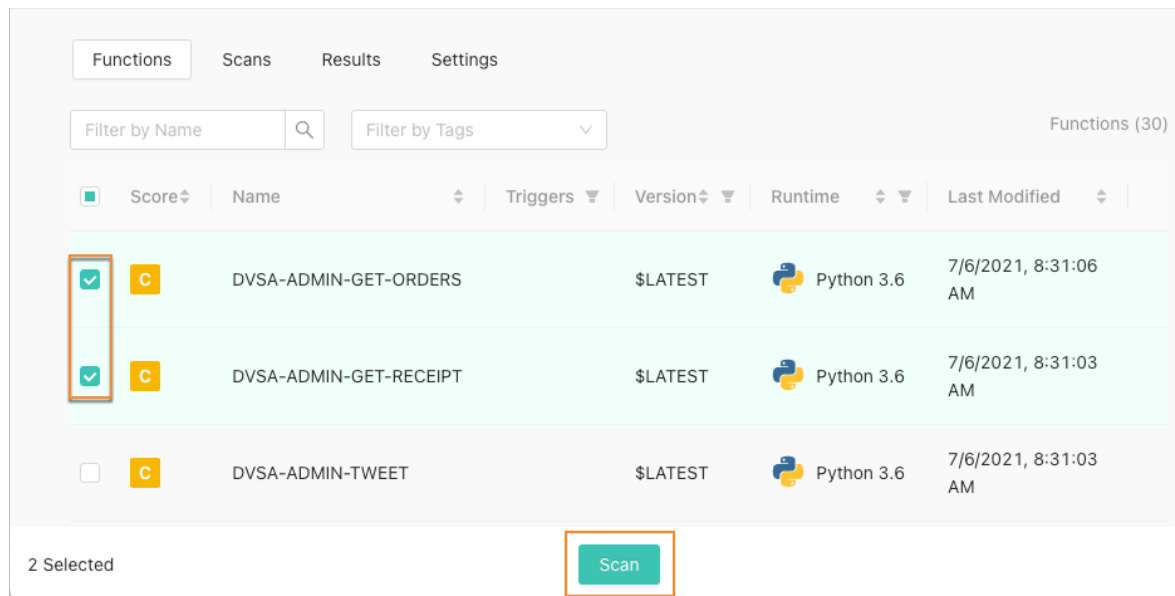
Although scans of all functions in your accounts occur automatically, you can also scan a specific function on demand.

## Before you begin

- Identify the functions that you want to scan.

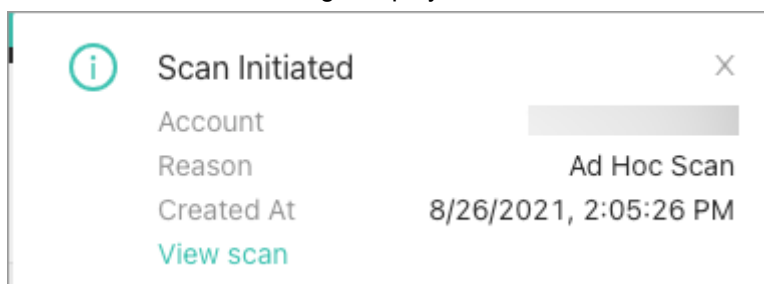
## Steps

- Select **Serverless** in the header.
- Select the account from the list.
- Select the check box next to one or more functions that you want to scan.



- Select **Scan**.
- In the Confirm Scan window, verify or change the scan type settings for the selected functions and select **OK**.

The **Scan Initiated** message displays.



- To view the scan results, select **View scan** in the Scan initiated message to view the scan results. Alternatively, select the **Scan** tab and select an **Ad hoc scan**.

If Contrast detects multiple occurrences of the same vulnerabilities in scanned functions, Contrast updates the existing reported vulnerabilities with new data (for example, timestamp or use), rather than creating new vulnerabilities.

## Functions table explained

The Functions table contains information about:

- Score:** The [contextual risk score \(page 988\)](#) for the function
- Name:** The function name
- Triggers:** The service that triggered the event
- Version:** The function version

- **Runtime:** The runtime language
- **Last Modified:** The last time the function was modified
- **Last Scanned:** The last execution time of the scan
- **Issues:** The items found during the scan that may or may not require attention. These results are generated from the Contrast Serverless and AWS Inspector sources. This displays only if you have Contrast Serverless and AWS Inspector support for the results.


## View results


View results to see details about vulnerabilities for permissions, dependencies, exploits, and CVEs.

## Before you begin

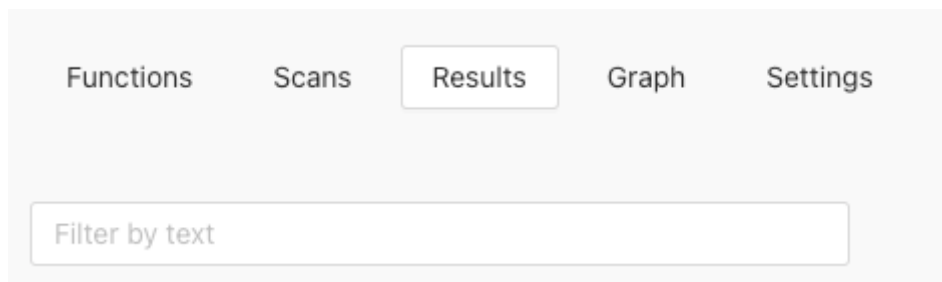
- Ensure that at least one scan is complete.

## Steps

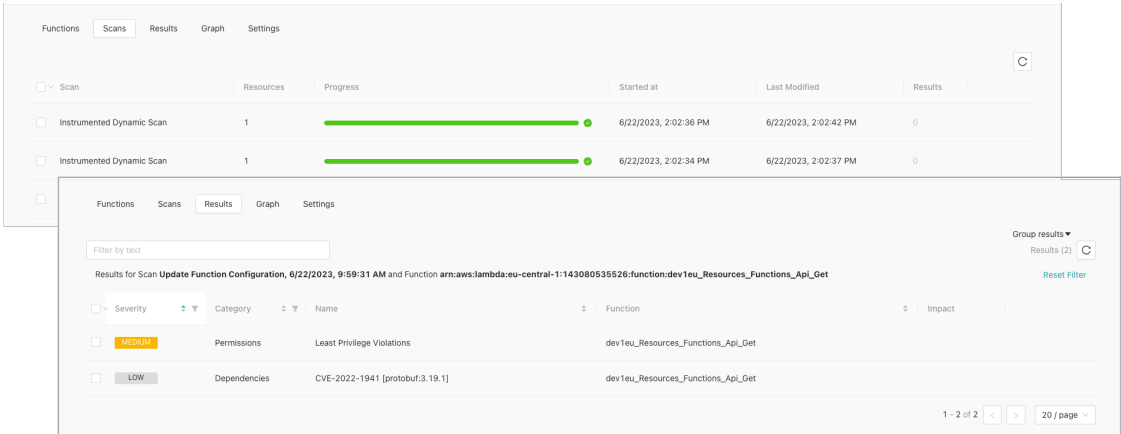
1. Select **Serverless** in the header.
2. To view results for all scans, select **Results**. See [Scan status details \(page 980\)](#) for more information about the meaning of the scan results.
3. To filter results in the Results tab, select the **Filter** icon (  ) next to the column headers.
  - a. The **Severity** is based on vulnerabilities in the application. See the [Application scoring guide \(page 1269\)](#) for information about the levels.
  - b. The **Category** is based on vulnerabilities in the type of function.
  - c. The **Function** is based on the function found in your account.  
Note that results can also be grouped by **Category** or **Function** by selecting the option under **Group results** on the upper-right side of the screen.
  - d. The **Source** is based on the platform providing the results. It is either from Contrast or AWS Inspector. Click the icon to view details about the results. Note that AWS Inspector results will display only if you have AWS Inspector support for your accounts.

To clear the filters, select the green **Filter** icon (  ) icon next to the column headers and select **Reset** in the filter window.

4. To search for a function in the Results tab, type a search term in the search field.



5. To view results for a single scan, from the Scans tab, select a scan row.  
This action shows a scan details page.
6. To view result details for a function:
  - a. From the Results tab, select a row in the list.
  - b. From the scan details page, select a number in the Results column for a function.



The results detail page looks similar to this example:

## Least Privilege Violations



MEDIUM

Category: Permissions | Function: `arn:aws:lambda:eu-central-1:143080535526:function:dev1eu_Resources_Functions_Api_Get` | ID: LEAST\_PRIVILEGE...

## Description

The Attached Role `arn:aws:iam::143080535526:role/dev1eu_Resources_Functions_Api_Get_Role` has an over permissive policy that may violate the principle of least privilege. For more information on AWS least privilege: [docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html#grant-least-privilege](https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html#grant-least-privilege)

## What happened

- `dev1euResourcesLambdaPolicy_V2-c743f03`: Unused permission `AddLayerVersionPermission` for service `lambda`
- `dev1euResourcesLambdaPolicy_V2-c743f03`: Unused permission `AddPermission` for service `sns`
- `dev1euResourcesLambdaPolicy_V2-c743f03`: Unused permission `CancelKeyDeletion` for service `kms`
- `dev1euResourcesLambdaPolicy_V2-c743f03`: Unused permission `CreateAlias` for service `kms`
- `dev1euResourcesLambdaPolicy_V2-c743f03`: Unused permission `CreateKey` for service `kms`
- `dev1euResourcesLambdaPolicy_V2-c743f03`: Unused permission `DeleteAlias` for service `kms`
- `dev1euResourcesLambdaPolicy_V2-c743f03`: Unused permission `DeleteMessage` for service `sqs`
- `dev1euResourcesLambdaPolicy_V2-c743f03`: Unused permission `DescribeEventBus` for service `events`
- `dev1euResourcesLambdaPolicy_V2-c743f03`: Unused permission `DescribeKey` for service `kms`
- `dev1euResourcesLambdaPolicy_V2-c743f03`: Unused permission `EnableKey` for service `kms`
- `dev1euResourcesLambdaPolicy_V2-c743f03`: Unused permission `GetBucketPolicy` for service `s3`
- `dev1euResourcesLambdaPolicy_V2-c743f03`: Unused permission `GetQueueAttributes` for service `sqs`
- `dev1euResourcesLambdaPolicy_V2-c743f03`: Unused permission `GetQueueUrl` for service `sqs`
- `dev1euResourcesLambdaPolicy_V2-c743f03`: Unused permission `GetTopicAttributes` for service `sns`
- `dev1euResourcesLambdaPolicy_V2-c743f03`: Unused permission `ListAliases` for service `kms`
- `dev1euResourcesLambdaPolicy_V2-c743f03`: Unused permission `PutBucketPolicy` for service `s3`
- `dev1euResourcesLambdaPolicy_V2-c743f03`: Unused permission `PutPermission` for service `events`
- `dev1euResourcesLambdaPolicy_V2-c743f03`: Unused permission `ReceiveMessage` for service `sqs`
- `dev1euResourcesLambdaPolicy_V2-c743f03`: Unused permission `RemoveLayerVersionPermission` for service `lambda`
- `dev1euResourcesLambdaPolicy_V2-c743f03`: Unused permission `RemovePermission` for service `events`
- `dev1euResourcesLambdaPolicy_V2-c743f03`: Unused permission `RemovePermission` for service `sns`
- `dev1euResourcesLambdaPolicy_V2-c743f03`: Unused permission `ScheduleKeyDeletion` for service `kms`

## Violating policies

`dev1euResourcesLambdaPolicy_V2-c743f03`[View violating policy](#)

## Remediation

Replace the existing policy with one of the following:

[JSON](#) [TERRAFORM](#) [YAML](#)

```
[
 {
 "PolicyName": "dev1euResourcesLambdaPolicy_V2-c743f03",
 "PolicyDocument": {
 "Version": "2012-10-17",
 "Statement": [
 {
 "Sid": "1",
 "Effect": "Allow",
 "Action": [
 "logs:CreateLogGroup",
 "logs:CreateLogStream",
 "logs:PutLogEvents"
],
 "Resource": [
 "arn:aws:logs:*:*:*"
]
 },
 {
 "Sid": "2",
 "Effect": "Allow",
 "Action": [
 "dynamodb:GetItem",
 "dynamodb:Query",
 "dynamodb:Scan",
 "dynamodb:PutItem",
 "dynamodb:UpdateItem",
 "dynamodb:BatchWriteItem",
 "dynamodb>DeleteItem"
],
 "Resource": [
 "arn:aws:dynamodb:eu-central-1:143080535526:table/dev1eu_resources.functions",
 "arn:aws:dynamodb:eu-central-1:143080535526:table/dev1eu_resources.resources",
 "arn:aws:dynamodb:eu-central-1:143080535526:table/dev1eu_resources.lookup",
 "arn:aws:dynamodb:eu-central-1:143080535526:table/dev1eu_resources.locks"
]
 }
]
 }
 }
]
```

## Result details

The result details displayed here depend on the category of a vulnerability.

All results include this information:

- **Description:** A description of the vulnerability.
- **What happened:** What occurred when the scan discovered the vulnerability
- **Remediation:** Steps to take to fix the vulnerability.

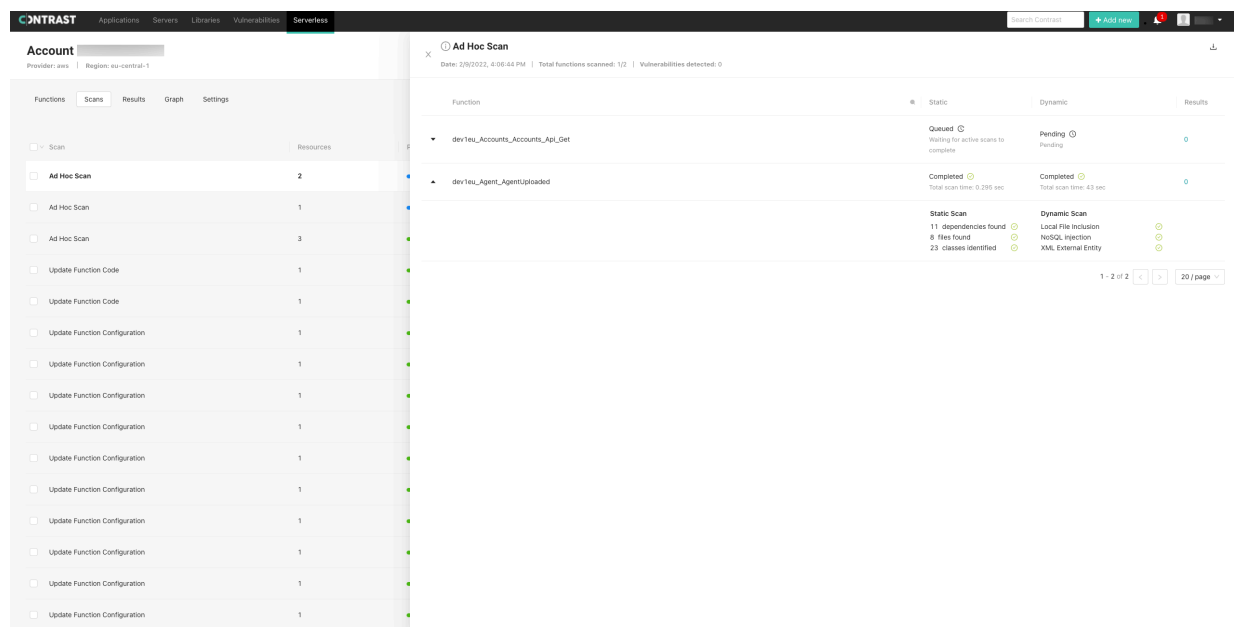
You might also see these details:

- **Violated policies:** Policies that the vulnerability violates.
- **Impact:** The area that the vulnerability affects, for example ses, s3, logs, or dynamodb.
- **Severity and metrics:** A severity score calculated to the vulnerability as well as metrics for the areas that the vulnerability impacts.

## Scan status details

Under the Serverless Scan tab, you can see what the system is scanning as well as the types of scans the functions are tested against.

Once on the **Scan** page, click the scan name to open the scan status details tab.



Here's a list of the static and dynamic scan statuses with their details.

Status	Description
Scanning.....	Scan in progress
Pending	Pending static scan results
Queued	Waiting for X active scans to complete Where X is the number of active scans. Scan is queued and will start soon.
Completed	Scan complete
Unsupported	Unsupported Lambda runtime The function runtime language is unsupported.
	Unsupported Lambda trigger
	The function has an unsupported trigger configuration OR no identified trigger configuration.
Excluded	Scan disabled in <a href="#">Settings (page 983)</a> To scan, change the Inventory settings or run an ad-hoc scan on the function.



Status	Description
Canceled	Newer scan initiated
	A newer version of the function is already in queue.
	Lambda state inactive
	Lambda reached limit of 5 layers
	Contrast cannot scan functions that already contain the maximum limit of 5 layers.
	Lambda scan already in progress
	Lambda last update status failed
Failed	Unable to verify agent
	Unable to decrypt/encrypt the environment variables
	Agent failure
	Contrast unable to invoke the Cloud Agent function OR scan failed during static analysis.
	Agent modified
	Misconfigured Lambda handler
	Parsing error

- Click the arrow icon to expand the details to view the dependencies, files, and classes found in the scan.
- Click the number under the Results column to open the [Results \(page 977\)](#) tab.

## Download serverless scan results

After a scan completes, you can download the results as a CSV file.

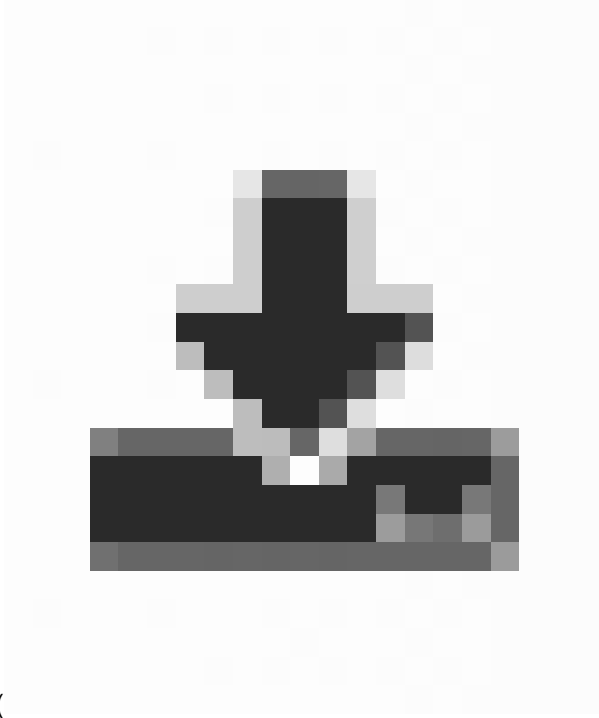
### Before you begin

- Identify the scan with results you want to download.

### Steps

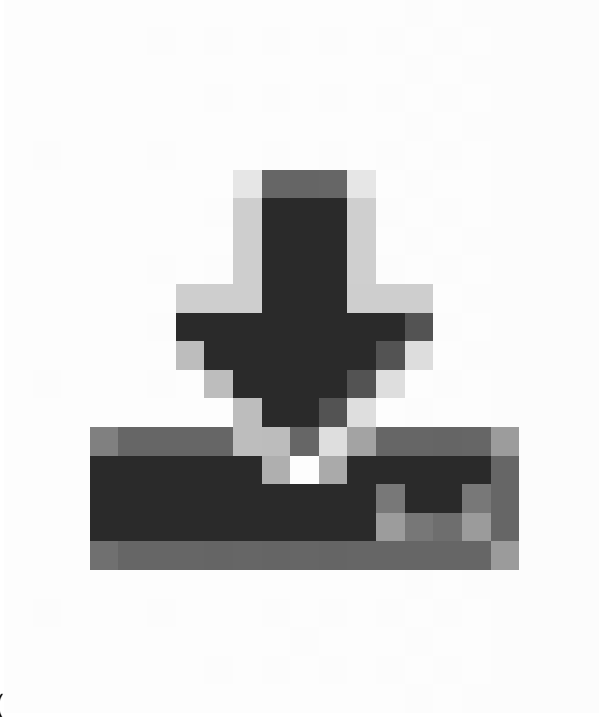
1. Select **Serverless** in the header.
2. Select either the **Scans** tab or the **Results** tab.
3. To download scan results from the **Scans** tab, hover over the end of a row for a scan and select the download icon.
4. To download scan results from the **Results** tab:

- a. Hover over the end of a row for a scan and select the download icon



OR

- b. Click the scan row to open the details page and then select the download icon



Results are downloaded as CSV files.

## Change inventory criteria

The inventory criteria that you specify determine the functions scanned by Contrast Serverless. By default, Contrast scans all functions discovered in your AWS account.

Excluding a function excludes it from the inventory and scan results.

## Before you begin

- Identify the functions you want to include or exclude in scans.

## Steps

- Select **Serverless** in the header.
- Select the **Settings** tab.
- Under Inventory, specify the criteria:
  - Include or exclude a tag associated with one or more functions. Optionally, specify a value for the tag to further refine the inventory.
  - Include or exclude functions by name. The options are: **Name is**, **Name starts with**, or **Name ends with**.

The screenshot shows the 'Settings' tab in the Contrast interface. Under the 'Inventory' section, there is a heading 'Contrast will discover all functions in your AWS account.' followed by the instruction 'To edit tags, names or environment variables that limit the scan inventory, edit the inventory criteria:'. Below this, there are two rows of input fields. The first row has a dropdown set to 'Include', a dropdown set to 'Tag', a text input with 'Build123', and a placeholder 'Tag value - leave empty for any'. The second row has a dropdown set to 'Exclude', a dropdown set to 'Name', a dropdown set to 'starts with', and a text input with 'cron'. At the bottom of the form, there is a green '+ Add criteria' link and a note: 'Any functions or results that no longer apply will be marked as inactive.'

- Select **Save and Rescan**.  
Contrast rescans the inventory automatically based on the new criteria.

## Change serverless scan settings

The scan settings affect the type of scan that Contrast Serverless performs on all functions.

You can [change these settings \(page 975\)](#) for a manual scan of selected functions.

## Before you begin

- Determine if you want to use static scans, dynamic scans, or both.

## Steps

- Select **Serverless** in the header.
- Select the **Settings** tab.
- Under **Scan**, select the types of scans that you want to use:
  - Static analysis:** This scan type looks at relevant static code and configuration assessments to discover new vulnerabilities.  
During a static scan, Contrast adds a Lambda function to your account. Once the scan completes, the function exits.
  - Dynamic analysis:** For AWS accounts only. This scan type looks at dynamic assessments based on the specific update introduced to the tested environment.  
During a dynamic scan, Contrast tries to send malicious input to the code and then exercises the code to discover vulnerabilities.  
For more information about the Instrumented Dynamic analysis option see [Scan types and monitoring \(page 967\)](#).

**IMPORTANT**

Serverless scans do not change your function code.

4. Select **Save**.

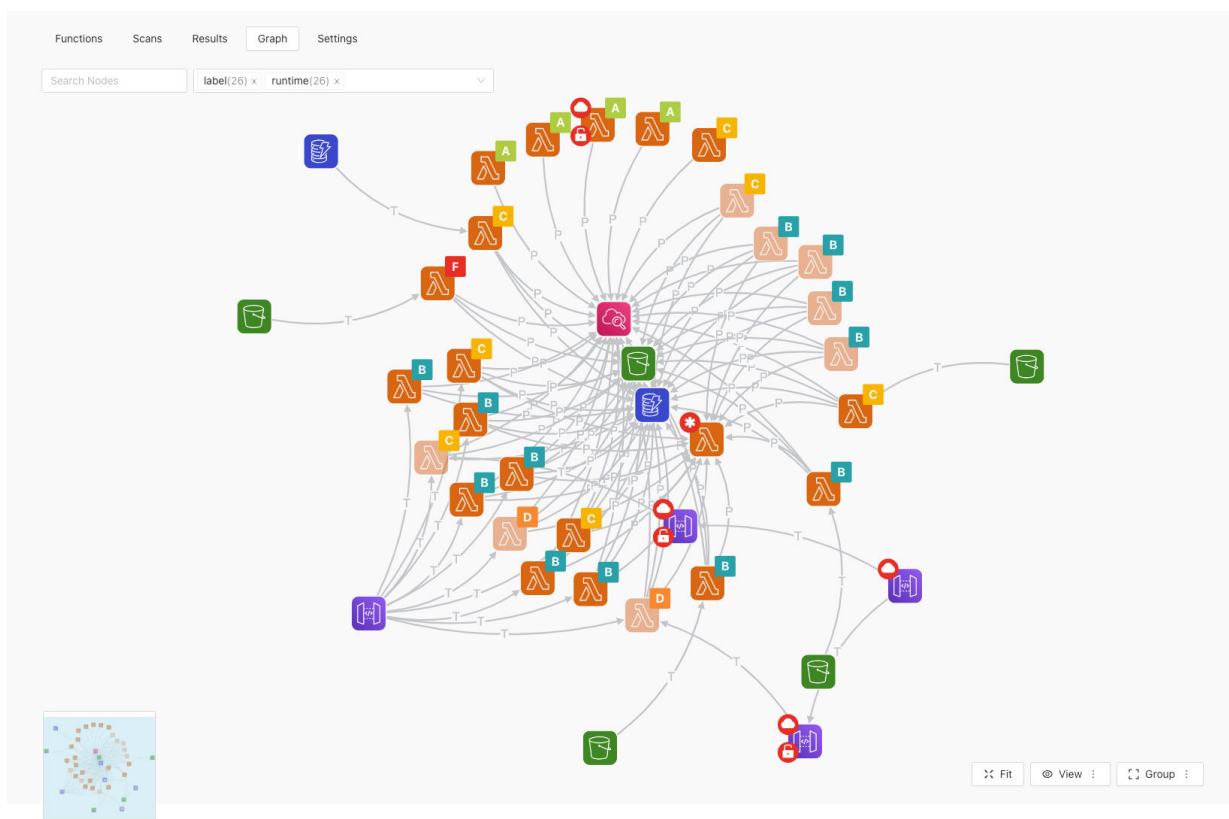
**View function and service relationships**

The Graph tab displays a diagram that shows the relationship between functions and services. You can also view details about each element in the diagram, including:

- Tags
- Vulnerability results
- Permissions
- Security posture score: The security posture score is based on the function's trigger configuration. Internet-accessible triggers and misconfigurations, such as open buckets and unauthenticated APIs, receive lower scores.
- Unused functions (shadow functions) from dynamic instrumented analysis

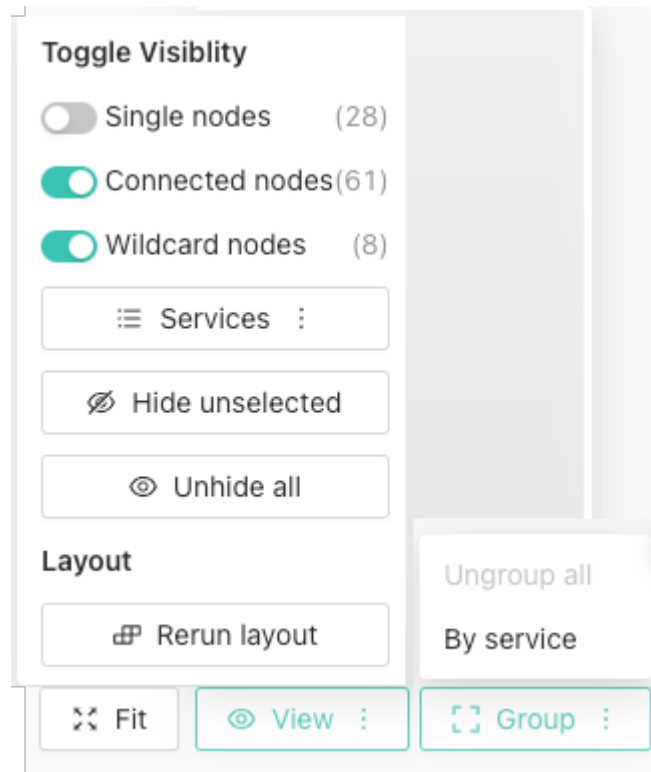
**WARNING**

There is a limit to the results available in this graph. See [Graph limits \(page 986\)](#) for more information.

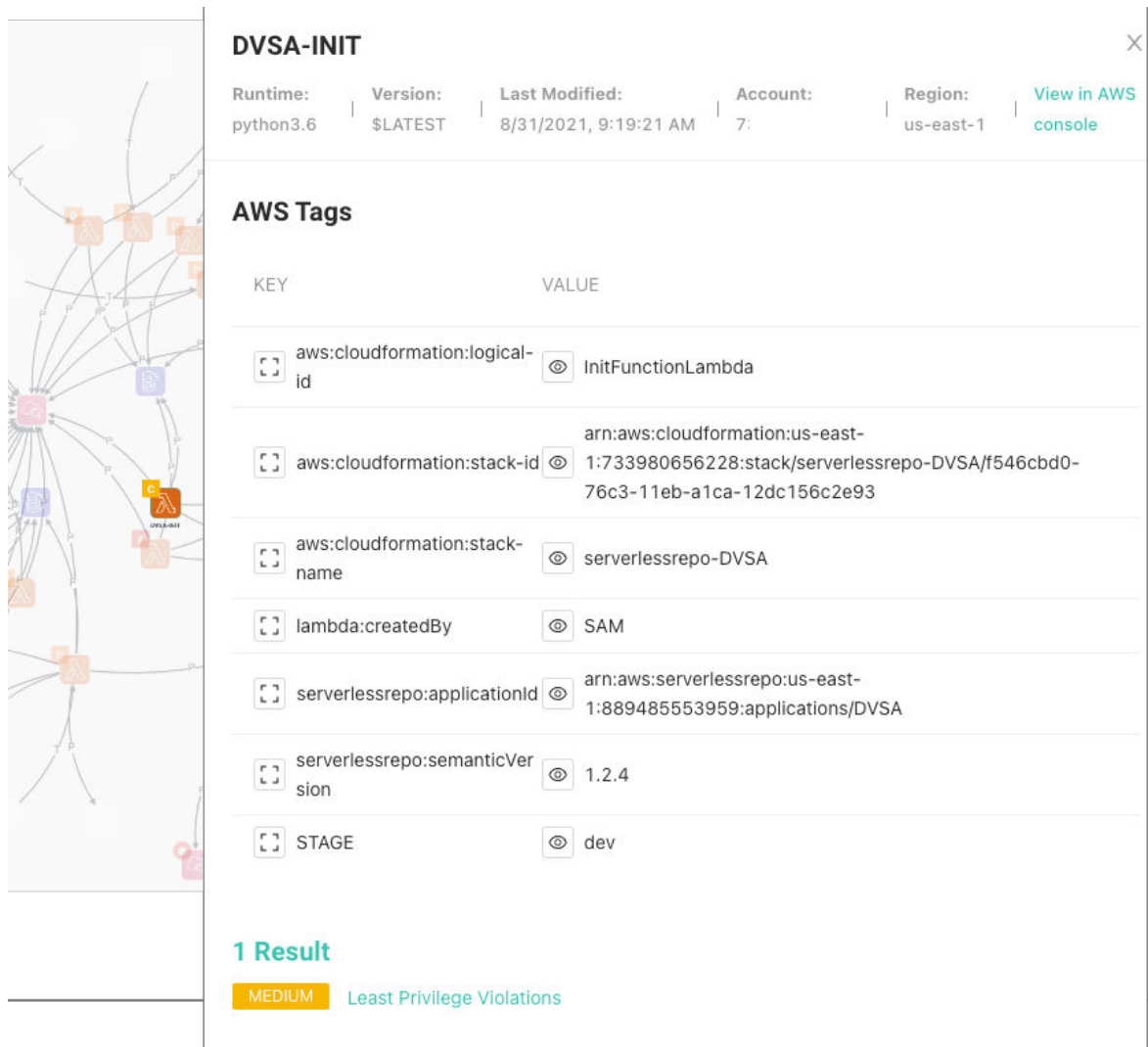


## Steps

1. Select **Serverless** in the header.
2. Select the **Graph** tab.
3. To adjust the view, use the options at the bottom of the graph.



- **Fit:** This allows you to resize the graph to fit your display
  - **View:** This allows you to filter the view by nodes and services. You can also hide or unhide nodes.
  - **Group:** You can group services together
4. Select the element to view details about individual elements.  
Contrast displays a details window for the selected element. Select the tags to filter the view.



**DVSA-INIT**

Runtime: python3.6 | Version: \$LATEST | Last Modified: 8/31/2021, 9:19:21 AM | Account: 7 | Region: us-east-1 | [View in AWS console](#)

**AWS Tags**

KEY	VALUE
aws:cloudformation:logical-id	InitFunctionLambda
aws:cloudformation:stack-id	arn:aws:cloudformation:us-east-1:733980656228:stack/serverlessrepo-DVSA/f546cbd0-76c3-11eb-a1ca-12dc156c2e93
aws:cloudformation:stack-name	serverlessrepo-DVSA
lambda:createdBy	SAM
serverlessrepo:applicationId	arn:aws:serverlessrepo:us-east-1:889485553959:applications/DVSA
serverlessrepo:semanticVersion	1.2.4
STAGE	dev

**1 Result**

**MEDIUM** Least Privilege Violations

## Graph limits

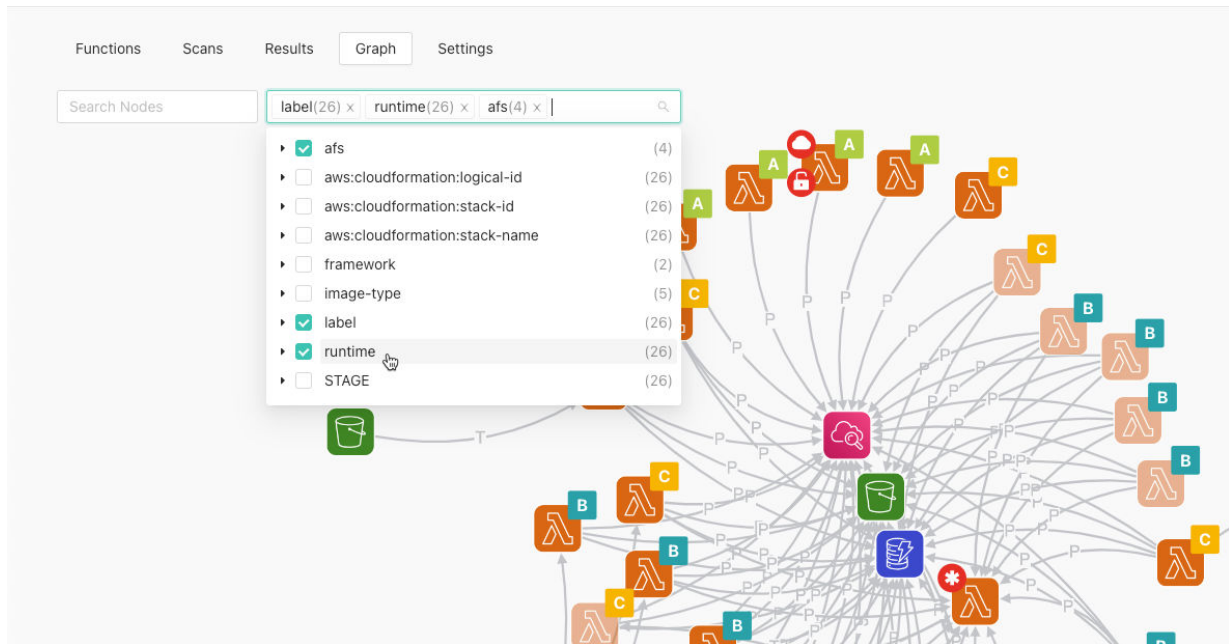
Due to performance considerations, we currently disable graph rendering when the element count exceeds a specific number of 8000.

### What is the element count?

Since this is a performance, rather than a product measure, the elements count is not the number of AWS/Azure resources but rather the number of renderable elements; this means all **Nodes** and **Edges** (connections). Therefore it is not possible to predict the specific number of elements that will result from rendering any single Function and its related components as the actual element number depends solely on the graph structure, which is unique to each environment.

### What can I do if I have too many elements and the graph does not render?

To allow you to still render the graph even if your account is too large, we added a filtering option (**tag filters**) using resource tags.



Using the filter will reduce the number of returned resources and will display only the functions that contain the selected tags and their related resources. This will help you stay under the element limit but if the selection is too large, you may run into the same problem. The tag selection works the same way as the **Functions** tab.



### IMPORTANT

The tag filter is currently the only approach that will actually limit the number of rendered elements that count against the limit. Search or hiding certain elements will have no effect.

## Adding/updating tags

If you have functions that do not have a tag assigned, you will not be able to see them using the tag filter. To properly utilize this feature, all your functions need to have tags assigned using a few different values so you can filter down to parts of your system. Potential tag ideas are `application`, `service`, `runtimeLanguage`, `team` or some other custom groupings depending on your needs.

Use the method that suits you best, either adding them directly in the AWS Console or using your IaC solution.

### When will I see the new tags in the graph?

While we do a near-immediate update of the **Functions** tab based on change events such as `TagResource`, this does not apply to the graph as we do not currently support partial updates and only run full scheduled discoveries. In most environments, this would be at 6 AM GMT. This means that you have to wait until the next discovery for the tags to appear.

CloudNative developers can also trigger the graph discovery prematurely using the `Resources_Utils_TriggerGraphDiscovery_V1` Lambda function.

Alternatively, changing [Inventory settings \(page 982\)](#) will trigger both **Resources** and **Graph** discoveries over the entire account. If the Inventory settings are empty, adding a dummy **Exclude** rule would be considered a change but would still include everything.

In summary, new tags will appear if either:

- Graph discovery runs daily on schedule at 6 AM GMT (some regions may have different time settings)
- `Resources_Utills_TriggerGraphDiscovery_V1` Lambda function is triggered with the specific organization ID/account ID
- Changes in Inventory settings

## Account Inventory Settings

During onboarding or later in **Settings**, you can change the **Inventory settings** to limit the scope of what you see in the system and what is scanned. Here you can use both **Function** tags and names.

However, keep in mind that modifying this will change the scope of the whole system, including what you see in **Functions** and what is actually scanned and protected. Depending on your use case it could also be a reasonable way to limit the size of the graph.

## Contextual risk scores

Contrast's contextual risk scoring system provides insight into where there are actual risk points and what should be prioritized. The function contextual score helps you gauge the general performance of each function.

<input type="checkbox"/> Score	Name
<input type="checkbox"/> <b>A</b>	DVSA-ORDER-GET
<input type="checkbox"/> <b>B</b>	DVSA-ORDER-BILLING
<input type="checkbox"/> <b>C</b>	DVSA-USER-CREATE
<input type="checkbox"/> <b>D</b>	DVSA-ADMIN-GET-ORDERS
<input type="checkbox"/> <b>F</b>	DVSA-ADMIN-ORDERS-MYSQL

Functions will have a letter grade **A** to **F**, which represents the overall posture of the function, and a numeric score from 35 to 100.

- **A**: 90 -100
- **B**: 80-89
- **C**: 70-79
- **D**: 60-69
- **F**: 35-59

The overall contextual Risk Score is calculated as follows: **Average = (Vulnerability score + Impact score + Likelihood score) / 3.**

## Vulnerability score

The vulnerabilities identified during the function scans (static and dynamic).



- Types of vulnerabilities:
  - Custom code exploits (Static and Dynamic)
  - Dependencies (CVEs)
  - Least Privilege violations
- To calculate the custom code vulnerability score, start with 100 points and subtract penalty points for the number of vulnerabilities found in a function multiplied by a penalty weight for their severity.
  - Critical: Multiply the number of vulnerabilities by 20
  - High: Multiply the number of vulnerabilities by 10
  - Medium: Multiply the number of vulnerabilities by 5
  - Low: Multiply the number of vulnerabilities by 1
  - For example: If the function has 0 Critical, 1 High, 0 Medium and 2 Low vulnerabilities, the score would be:  $100 - (20 \times 0) - (10 \times 1) - (5 \times 0) - (1 \times 2) = 88$

### Impact (access level) score

The permission (IAM roles) given to the function. The more permissions the function has, the higher the risk.

To calculate the impact score, we inspect and score each of the 5 permission categories: List, Read, Write, Tagging, and Permissions Management for each service. Then, we start with 100 points and subtract penalty points for the access level of each service.

For example, given the following IAM policy:

```
{
 "Effect": "Allow" ,
 "Action": [
 "s3:GetObject",
 "sqs:*"
],
 "Resource": "*"
}
```

The score for each service access level would be calculated as the following:

```
{
 "s3": {
 "Read": 6,
 "Write": 3,
 "List": 3,
 "Tagging": 1,
 "Permissions management": 12
 },
 "sqs": {
 "Read": 3,
 "Write": 3,
 "List": 1,
 "Tagging": 1,
 "Permissions management": 6
 }
}
```

The overall score would be calculated as follows:

- s3: [6], sqs: [3,3,1,1,6] -->  $100 - (6+3+3+1+1+6) = 80$

## Likelihood (accessibility) score

The likelihood of an attacker reaching the function is based on the function trigger configuration.

Each service has a different score based on the ability of attackers to access the function as well as based on the trigger configuration (for example, authenticated/unauthenticated).

For example:

If the function has an EventBus set as a trigger, the chances for a potential attacker to access the Lambda function would be lower than accessing a Lambda with an API Gateway set as a trigger. Moreover, if the API Gateway is configured without any authentication (i.e., Open), then the function can be accessible by anyone, anywhere.

So the likelihood score for a function:

- with an EventBus as a trigger would be: 90
- with an (authenticated) API Gateway as a trigger would be: 75
- with an unauthenticated API Gateway as a trigger would be: 5 (the lowest possible score)
- without a trigger would be: 100 (the highest possible score)

## Upgrade Contrast Serverless

Contrast Serverless is set to auto-update in most cases. If there is a need to manually upgrade you can follow the steps listed on this page.

### Before you begin

- Create a role/user with the minimum required policies. See [this example \(page 969\)](#) for how to set it up.

### Steps

1. If you have not made any manual changes to the Contrast Serverless stack before your previous deployment, simply [uninstall \(page 994\)](#) the current Contrast Serverless stack.  
If you made manual changes to the previous Contrast Serverless stack, it is recommended that you [create a stack change-set \(page 992\)](#) and then continue with the following steps.
2. Click **Add New** in the toolbar.
3. Download a new template from Contrast by selecting **Download CFT**.

## Let's get started

Create a new stack based on these defaults, or specify function criteria.

### Cloud provider

Select the cloud provider that holds your account

AWS

Azure

### Inventory

Contrast will discover all functions in your AWS account.

To specify tags, names or environment variables that limit the scan inventory, add criteria:

+ Add Criteria

### Initial scan

Scan all functions in your account, then continuously scan all changes to identify:

☒ Functions matching inventory criteria

#### ☒ Static analysis

☒ Least privilege - Detecting unused permissions

☒ CVEs - Detecting vulnerable dependencies

☒ SAST - Detecting custom-code vulnerabilities

☒ Malware - Detecting malicious files

☐ Dynamic analysis - Fuzzing custom-code vulnerabilities (invokes functions)

### Deployment

Deploy with a new stack in AWS, or download the CFT to use in your pipeline.

Create new stack



Download CFT

4. Select either JSON or YAML.
5. Update the stack.

Update stack

Prerequisite - Prepare template

Prepare template

Every stack is based on a template. A template is a JSON or YAML file that contains configuration information about the AWS resources you want to include in the stack.

☐ Use current template

☒ Replace current template

☐ Edit template in designer

Specify template

A template is a JSON or YAML file that describes your stack's resources and properties.

Template source

Selecting a template generates an Amazon S3 URL where it will be stored.

☐ Amazon S3 URL

☒ Upload a template file

Upload a template file

Choose file

ContrastCloudNative\_CFT.yaml

JSON or YAML formatted file

S3 URL: https://s3.eu-central-1.amazonaws.com/cf-templates-1mtxj8diu6zvc-eu-central-1/2023-01-24T164553.73520xv-ContrastCloudNative\_CFT.yaml

View in Designer

Cancel

Next

6. Click **Submit** on the resulting screen.

Use Contrast

991

Capabilities

**The following resource(s) require capabilities: [AWS::IAM::ManagedPolicy]**

This template contains Identity and Access Management (IAM) resources that might provide entities access to make changes to your AWS account. Check that you want to create each of these resources and that they have the minimum required permissions. [Learn more](#)

☒ I acknowledge that AWS CloudFormation might create IAM resources.

[View change set](#) [Cancel](#) [Previous](#) [Submit](#)

## Stack change-set

To create a stack change-set, follow these steps.

### Before you begin

- Identify Contrast Serverless stack on the AWS CloudFormation console.

### Steps

- Log in to your Contrast dashboard and download a new template from Contrast by selecting **Download CFT**.

#### Let's get started

Create a new stack based on these defaults, or specify function criteria.

**Cloud provider**

Select the cloud provider that holds your account

[AWS](#) [Azure](#)

**Inventory**

Contrast will discover all functions in your AWS account.

To specify tags, names or environment variables that limit the scan inventory, add criteria:

[+ Add Criteria](#)

**Initial scan**

Scan all functions in your account, then continuously scan all changes to identify:

☒ Functions matching inventory criteria

☒ **Static analysis**

- ☒ **Least privilege** - Detecting unused permissions
- ☒ **CVEs** - Detecting vulnerable dependencies
- ☒ **SAST** - Detecting custom-code vulnerabilities
- ☒ **Malware** - Detecting malicious files

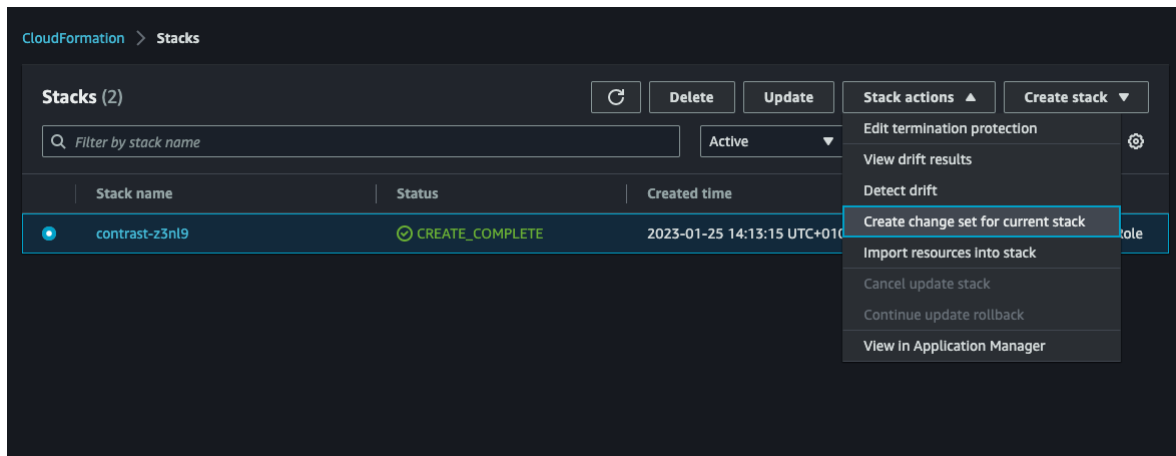
☐ Dynamic analysis - Fuzzing custom-code vulnerabilities (invokes functions)

**Deployment**

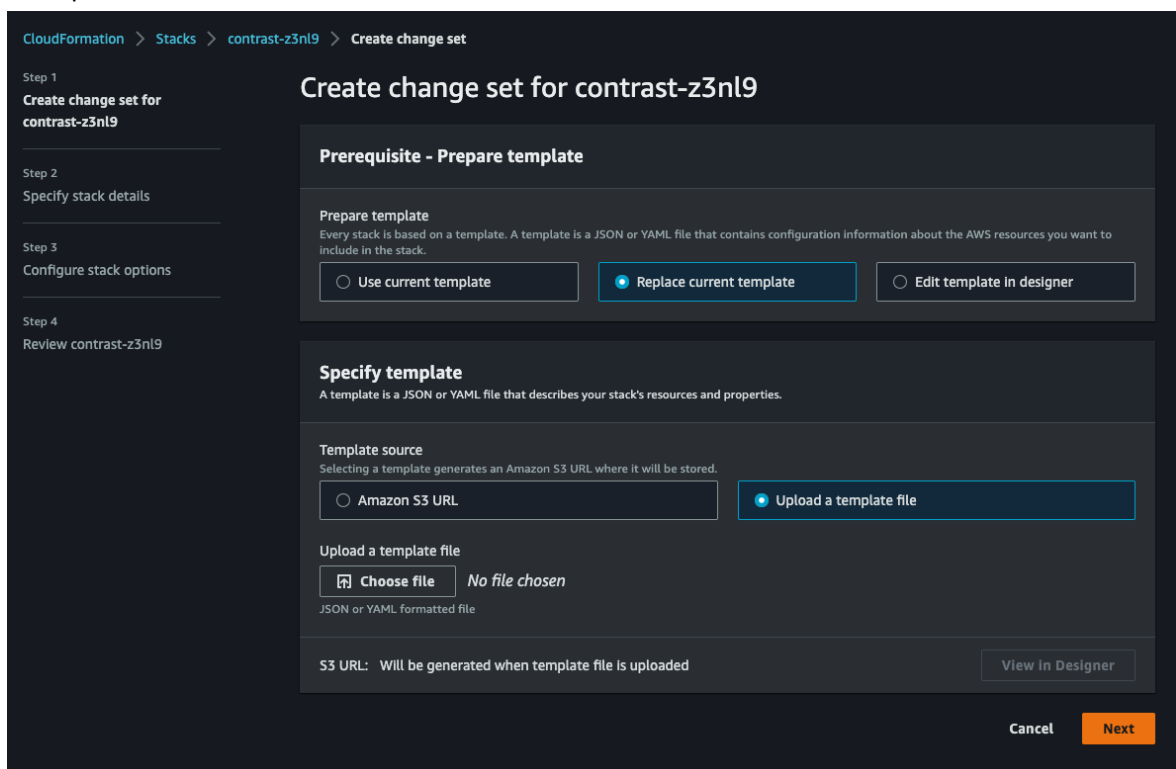
Deploy with a new stack in AWS, or download the CFT to use in your pipeline.

[Create new stack](#) [Download CFT](#)

- Select either JSON or YAML.
- Log in to your AWS account or use the AWS CLI/API.
- Select **Contrast Serverless Stack**.
- Select **Stack actions** > **Create change set for current stack**.



6. Select **Replace current template** > **Upload a template file** and upload the template downloaded in step 1.



## Block accounts

You can block all Contrast Serverless activities for an account.

1. Select **Serverless** in the header.
2. Select the **Settings** tab.
3. Scroll to the bottom of the page and click **Block Account**.

This will block all activities including automatic and user-requested scans and function updates.



### NOTE

Contact [Contrast Support](#) to unblock accounts.

## Offboard Contrast Serverless

Offboarding a Contrast Serverless account from Azure requires the use of a script to remove the deployment.

### Before you begin

- Identify the objects that you want to remove

### Steps

- Select **Serverless** in the header.
- Select the account to be offboarded.
- Select the **Settings** tab.
- Scroll down to the **Offboard Account** section and copy the script.
- Execute in a shell with an authenticated `az` command. For example, Azure CloudShell in Bash mode.

## Uninstall Contrast Serverless

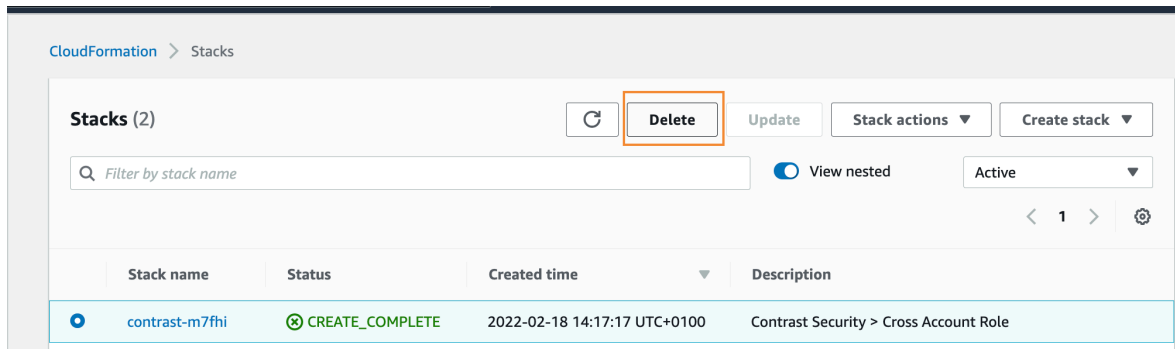
Uninstalling Contrast Serverless requires the deletion of a stack in the AWS console.

### Before you begin

- Identify the objects that you want to remove

### Steps

- Log in to your AWS account or use the AWS CLI/API.
- Continue with the steps for deleting a stack. See [AWS Stack deletion](#).



## Contrast CLI

Contrast CLI delivers SCA, SAST, and serverless capabilities at the command line. You can use the CLI locally or automate it in your CI/CD pipeline.

### Before you start

A few things to keep in mind:

- Enterprise Contrast users can [install \(page 995\)](#) and then start using the Contrast CLI.
- The legacy Contrast CLI will be deprecated as of October 2022.

### About Contrast CLI

Contrast Security brings security testing right to the developer's laptop. Make code and serverless security simple and efficient with quick scan times, market-leading accuracy, actionable results, and seamless integration.

Contrast CLI delivers:

- The fastest and most accurate SAST scanner
- Immediate and actionable results - scan code and serverless environments
- SCA capabilities by showing dependencies between open-source libraries, including where vulnerabilities were introduced

Start by [installing the Contrast CLI \(page 995\)](#).

## Contrast CLI supported languages and package managers

Contrast CLI supports these languages when using the `audit` command.

Package manager	Language	Files required	Notes
Composer	PHP	<i>composer.json</i> and <i>composer.lock</i>	
PyPI	Python	<i>pipfile</i> and <i>pipfile.lock</i>	
NuGet	.NET Core .NET Framework	MSBuild 15.0 or greater and <i>packages.lock.json</i> .	If the <i>packages.lock.json</i> file is unavailable it can be generated by setting <code>RestorePackagesWithLockFile</code> to true within each <i>*.csproj</i> file and running <code>dotnet build</code>  .NET is only supported using the <code>--legacy</code> option
RubyGems	Ruby	<i>gemfile</i> and <i>gemfile.lock</i>	
Maven Central	Java	<i>pom.xml</i>	Maven build platform including the dependency plugin
		<i>build.gradle</i>	gradlew dependencies or <code>./gradlew dependencies</code>
npm  This includes client-side and server-side JavaScript packages.	JavaScript	<i>package.json</i> and a lockfile (either <i>package-lock.json</i> or <i>yarn.lock</i> )  V2 and V3 lock files are supported	
Gopm	Go	<i>go.mod</i>	

## Install Contrast CLI

Use these procedures to install Contrast CLI.

### With Homebrew

Install from Contrast's tap with Homebrew by running the following commands.

```
brew tap contrastsecurity/tap
brew install contrast
```

### With NPM/YARN



#### NOTE

Currently, only Node.js 20.18.3+ is supported.

Install using npm or yarn `@contrast/contrast`.

Use this command:

```
npm install --location=global @contrast/contrast@2
```

## With Binaries

- Go to [artifacts](#). If you prefer to automate using the latest version of the CLI, point to [latest-version.txt](#).
- Download the latest package.
- You must allow `execute` permissions on the file depending on your OS.

Once installed, continue by [authenticating your account \(page 996\)](#).

## Authenticate your credentials

Authenticate to store your credentials before scanning for vulnerabilities.

Run the following `auth` command to store your credentials locally.

```
contrast auth
--api-key <your API key>
--authorization <your authorization header>
--host <your host domain>
--organization-id <your organization ID>
```

In Contrast, under **user menu > User settings > Profile**, locate and copy this information:

- API key
- Organization ID
- Authorization header

You will also need your Contrast URL for the `--host` line.

Once authenticated, [perform an analysis \(page 996\)](#).

## Security analysis

Use Contrast CLI to perform security analysis.

## Run a SAST scan

1. In the terminal, type the following code: `contrast scan -f <file name>`.
2. In the results click the link to view the [scan results \(page 887\)](#).

## Find vulnerable libraries

1. In the terminal, type the following code: `contrast audit`.
2. If you used the `--track` flag with the audit command, click the link in the results to open the [library view \(page 924\)](#).

## Find vulnerabilities in your AWS lambda functions

1. In the terminal, type the following code: `contrast lambda--function-name [option]`.
2. In the results, review any recommendations and update policies based on the provided information.



## Find vulnerabilities with Contrast Assess

1. Install or update a Contrast agent:
  - [Use Assess CLI with Java agents \(page 997\)](#)
  - [Use Assess CLI with .NET agents \(page 998\)](#)
  - [Use Assess CLI with Node.js agents \(page 999\)](#)
  - [Use Assess CLI with Python agents \(page 1000\)](#)
  - [Use Assess CLI with Ruby agents \(page 1000\)](#)
  - [Use Assess CLI with Go agents \(page 1001\)](#)
2. In the terminal, type the following code: `contrast assess`  
This command generates the agent configuration file that the Contrast CLI and the agent share.  
The default locations for the configuration file are:
  - **MacOS and Linux**  
`/etc/contrast/contrast_security.yaml`
  - **Windows**  
`%ProgramData%\Contrast\contrast_security.yaml`You have the option of specifying a different location with `--config-path`.



### NOTE

If your user does not have write permissions to the directory where the configuration file is located, use `sudo` or a similar mechanism to create the folder. For example:

```
sudo mkdir /etc/contrast
```

Then, grant all users read and write permissions. For example:

```
sudo chmod 777 /etc/contrast
```

3. Run your application in your IDE or a second terminal window.
4. Exercise your application, either interactively or using automated API or end-to-end tests.
5. View the results in the terminal where you entered the Contrast Assess CLI command.

## Use Assess CLI with Java agents

Use this procedure if you are using Contrast Java agents and want to use the CLI to find vulnerabilities while running API or end-to-end testing.

### Before you begin

- Verify your application can use the Assess CLI by checking the [Java supported technologies \(page 120\)](#).

### Steps

1. To install the latest Java agent, download it from [Maven Central](#).



### IMPORTANT

Do not create a configuration (YAML) file for the agent. The Assess CLI generates this file automatically.

2. Open a terminal window and enter the Assess CLI command:

```
contrast assess
```

This command generates the agent configuration file that the Contrast CLI and the agent share. [CLI commands \(page 1002\)](#) describes the options for this command, including the path for the configuration file.

You see output similar to this:

```
Configuration file found at "user_path"
Waiting for the session to be created.
```

3. In your IDE or a second terminal window, run your application with this command:

```
java -javaagent:<YourContrastJarPath> -jar <AppName>.jar
```

Alternative methods:

- **IntelliJ:** Modify the run configuration to include the following command as a VM argument:

```
-javaagent:<YourContrastJarPath>
```

Replace `<YourContrastJarPath>` with the path for the Java agent's `contrast.jar` file.

Using the updated run configuration automatically runs your Java application with the Contrast agent.

- **VS code:** Modify `vmArgs` setting in your launch configuration to include the following command as a VM argument:

```
-javaagent:<YourContrastJarPath>
```

Replace `<YourContrastJarPath>` with the path for the Java agent's `contrast.jar` file.

Add the agent under the `vmArgs` setting

4. Exercise your application, either interactively or using automated API or end-to-end tests.
5. View the results in the terminal window where you entered the Assess CLI command.

## Use Assess CLI with .NET agents

Use this procedure if you are using Contrast .NET agents and want to find vulnerabilities while running API or end-to-end testing.

### Before you begin

- Verify that your application can use the Assess CLI by checking the [.NET Core supported technologies \(page 271\)](#) or [.NET Framework supported technologies \(page 212\)](#).

### Steps

1. Install or update your agent manually.
  - [.NET Core \(page 273\)](#)
  - [.NET Framework \(page 214\)](#)



#### IMPORTANT

Do not create a configuration (YAML) file for the agent. The Assess CLI creates this file automatically.

2. Open a terminal window and enter the Assess CLI command.

```
contrast assess
```

This command generates the agent configuration file that the Contrast CLI and the agent share. [CLI commands \(page 1002\)](#) describes the options for this command, including the path for the configuration file.

You see output similar to this:

```
Configuration file found at "user_path"
Waiting for the session to be created.
```

3. Run your application using your IDE or a second terminal window.
4. Exercise your application, either interactively or using automated API or end-to-end tests.
5. In the terminal window where you entered the Assess CLI command, view the results.

## Use Assess CLI with Node.js agents

Use this procedure if you are using Contrast Node.js agents and want to find vulnerabilities while running API or end-to-end testing..

### Before you begin

- Verify your application can use the Assess CLI by checking the [Node.js supported technologies \(page 332\)](#).
- Contrast Assess is intended for server-side applications only. Assess does not detect vulnerabilities in client-side code.
- The Node.js agent can only instrument JavaScript applications. If you are using TypeScript for your server-side code, [transpile it to JavaScript \(page 390\)](#).

### Steps

1. Install the latest version of the agent from the application's root directory with this command:

```
npm install @contrast/agent
```

If you want to use yarn, use this command:

```
yarn add @contrast/agent
```



#### IMPORTANT

Do not create a configuration (YAML) file for the agent. The Assess CLI creates this file automatically.

2. Open a terminal window and enter the Assess CLI command.

```
contrast assess
```

This command generates the agent configuration file that the Contrast CLI and the agent share. [CLI commands \(page 1002\)](#) describes the options for this command, including the path for the configuration file.

You see output similar to this:

```
Configuration file found at "user_path"
Waiting for the session to be created.
```

3. In your IDE or a second terminal window, run your application with a command similar to this one:

```
node -r @contrast/agent <server.js>
```

Replace `<server.js>` with your Node.js application's server start command. Adjust the command based on your application specifics.

This command requires the Contrast agent for Node.js and instruments your application's source code as it is read by the Node.js engine.

4. Exercise your application, either interactively or using automated API or end-to-end tests.
5. View the results in the terminal window where you entered the Assess CLI command.

## Use Assess CLI with Python agents

Use this procedure if you are using Contrast Python agents and want to find vulnerabilities while running API or end-to-end testing..

### Before you begin

- Verify your application can use the Assess CLI by checking the [Python supported technologies \(page 410\)](#).

### Steps

1. Install the agent using `pip`:

```
pip install contrast-agent
```



#### TIP

If you have a `requirements.txt` file, you can add `contrast-agent` to that file, and install with `pip install -r requirements.txt`.



#### IMPORTANT

Do not create a configuration (YAML) file for the agent. The Assess CLI creates this file automatically.

2. Verify that `autoconf` is installed on the system where you will run the agent.
3. Open a terminal window and enter the Assess CLI command:

```
contrast assess
```

This command generates the agent configuration file that the Contrast CLI and the agent share. [CLI commands \(page 1002\)](#) describes the options for this command, including the path for the configuration file.

You see output similar to this:

```
Configuration file found at "user_path"
Waiting for the session to be created.
```

4. Run your application using your IDE or a second terminal window.
5. Exercise your application, either interactively or using automated API or end-to-end tests.
6. In the terminal window where you entered the Assess CLI command, view the results.

## Use Assess CLI with Ruby agents

Use this procedure if you are using Contrast Ruby agents and want to find vulnerabilities while running API or end-to-end testing..

### Before you begin

- Verify your application can use the Assess CLI by checking the [Ruby supported technologies \(page 469\)](#).

## Steps

1. Add this entry to your gemfile:

```
gem 'contrast-agent'
```

2. Install or update your agent:

- Install the agent with this command:

```
bundle install
```

- Update the agent with this command:

```
bundle update contrast-agent
```



### IMPORTANT

Do not create a configuration (YAML) file for this agent. The Assess CLI creates this file automatically.

3. Configure middleware (Grape, Rails, or Sinatra)

- **Grape:** Add the middleware directly to your application class extending the `Grape::API` or to your `config.ru` file if a class is not available.

```
require 'contrast-agent'
use Contrast::Agent::Middleware, true
```

- **Rails:** No code change required.
- **Sinatra:** Add the middleware directly to your application class extending the `Sinatra::Base` or to your `config.ru` file if a class is not available.

```
require 'contrast-agent'
use Contrast::Agent::Middleware, true
```

4. Verify that autoconf is installed on the system where you will run the agent.
5. Open a terminal window and enter the Assess CLI command:

```
contrast assess
```

This command generates the agent configuration file that the Contrast CLI and the agent share. [CLI commands \(page 1002\)](#) describes the options for this command, including the path for the configuration file.

You see output similar to this:

```
Configuration file found at "user_path"
Waiting for the session to be created.
```

6. Run your application using your IDE or a second terminal window.
7. Exercise your application, either interactively or using automated API or end-to-end tests.
8. In the terminal window where you entered the Assess CLI command, view the results.

## Use Assess CLI with Go agents

Use this procedure if you are using Contrast Go agents and want to find vulnerabilities while running API or end-to-end testing.

Running an application with the Go agent is different than most other Contrast agents. The Go agent is injected into the application's source code at compile time.

## Before you begin

- You can use the [Contrast Go Test Bench](#) application to test the Assess CLI.
- Verify your application can use the Assess CLI by checking the [Go supported technologies \(page 530\)](#).

## Steps

1. Open a terminal window and install the [Contrast Go agent \(page 531\)](#) in your environment (version 1.19 minimum),



### IMPORTANT

Do not create a configuration (YAML) file for the agent. The Assess CLI creates this file automatically.

2. Verify that the compiler is installed using this command:

```
go version
go version go1.19.1 darwin/arm64
```

3. Install, compile, and run your application.

To verify that the application is running without Contrast implementation, open a browser and navigate to the application. Enter CTRL-C to stop the application.

For example, if you are using the Contrast Go Test Bench application, you would navigate to localhost:8080.

4. Enter the Assess CLI command:

```
contrast assess
```

This command generates the agent configuration file that the Contrast CLI and the agent share. [CLI commands \(page 1002\)](#) describe the options for this command, including the path for the configuration file.

You see output similar to this:

```
Configuration file found at "user_path"
Waiting for the session to be created.
```

5. In your IDE or in a second terminal window, compile and run your application, which is now instrumented with the Contrast Go agent.

For example: if you are using the Contrast Go Test Bench application, the commands would look like this:

```
go-test-bench on main [!?] via v1.19.1 took 1h52m1s
contrast-go run ./cmd/gin/app.go
```

6. Open a third terminal window and exercise your application, either interactively or using automated API or end-to-end tests.

For example, if you are using the Contrast Go Test Bench application, the commands would look like this:

```
go-test-bench on main [!?] via v1.19.1 took 4s
go run ./cmd/exercise
```

7. In the first terminal window that you opened, view the results.

## Contrast CLI commands

The following is a listing of the commands available to basic and advanced users of Contrast CLI.

**Usage:**contrast [command] [options]

## Authentication/connectivity

### auth

If you already have a Contrast account, run the following `auth` command to store your credentials locally.

- **Usage:** `contrast auth`

```
contrast auth
--api-key <your API key>
--authorization <your authorization header>
--host <your host domain>
--organization-id <your organization ID>
```

You can then [run an analysis \(page 996\)](#) with the commands.

### config

Displays stored credentials.

- **Usage:** `contrast config`

Example:

```
contrastuser@userc-C02GD0LUMD6TTY ~ % contrast config
{
 version: '1.0.24',
 host: 'https://app.contrastsecurity.com',
 apiKey: 'wwEHMnYEIAujE03fFGH',
 organizationId: '0fde1b36-6986-4a14-b16d-6258aa913e5bceerfj',
 authorization:
 'Z211bG1hbmEubWYyaWFuaUBjb250cmFzdHNlY3VyaXR5LmNvbTpDUktMUTE3T1czMDU2NjlLO
 PDS',
 numOfRuns: 0
}
```

- **Options:**
  - `-c, --clear`  
Removes stored credentials.

### version

Displays Contrast CLI version.

- **Usage:** `contrast version`

Example:

```
contrastuser@usercsa-C02GD0LUMD6TTY ~ % contrast version
1.0.24
```

## Main functions

### audit

Searches for a dependency configuration file in the working directory to perform a security audit of dependencies and returns the results.

- **Usage:** `contrast audit [option]`

Example:

```
contrastuser@usercsa-C02GD0LUMD6TTY ~ % contrast audit
Searching for package manager files from /Users/contrastuser/Documents/
```

```
Contrast SCA audit started...
Contrast audit complete

Found 4 vulnerable libraries with 4 CVEs

CONTRAST-001 - [CRITICAL] minimist-1.2.5 introduces 1 vulnerability
 Issue: 1 Critical
 [C]CVE-2021-44906
 Advice: Update to version 1.2.6

CONTRAST-002 - [CRITICAL] json-schema-0.2.3 introduces 1 vulnerability
 Issue: 1 Critical
 [C]CVE-2021-3918
 Advice: Update to version 0.4.0

CONTRAST-003 - [HIGH] glob-parent-5.1.1 introduces 1 vulnerability
 Issue: 1 High
 [H]CVE-2020-28469
 Advice: Update to version 5.1.2

CONTRAST-003 - [HIGH] ansi-regex-0.2.1 introduces 1 vulnerability
 Issue: 1 High
 [H]CVE-2021-3807
 Advice: Update to version 6.0.1
```

- **Options:**

- **--fail**  
Fail a build based on the severity of CVEs found. Use with the **--severity** flag. For example, **contrast audit --fail --severity high**. Returns all failures if no severity level is specified. If a failure is detected the CLI will exit with code 2.
- **--file**  
Specify a directory or the file where dependencies are declared. (By default, Contrast CLI will search for project files in the current directory.) If multiple project files are found in the directory, you will be prompted to confirm the file to audit.  
**Alias:** -f
- **--help**  
Displays usage information for all `audit` command options.
- **--ignore-dev**  
Excludes developer dependencies from the results. All dependencies are included by default.  
**Alias:** -i
- **--legacy**  
Creates an application in Contrast (a legacy workflow). It displays a [dependency tree \(page 934\)](#) for your piece of code and utilizes metadata. Note that this is only available for Contrast CLI V2.0 and later.
- **--name**  
Set a custom project name. If the name is already in use, it will replace the results for that project. Avoid special characters.
- **--resource-group**  
Specify a resource group when generating SCA projects. Resource groups specify the applications, projects, and organization settings that users can access, based on their assigned roles. See [Resource groups \(page 1284\)](#) for more information.
- **--save**  
Generate and save an SBOM (Software Bill of Materials). Valid options are: **--save cyclonedx** and **--save spdx** (CycloneDX is the default format.).



**Alias: -s**

- **--severity**

Specify the minimum severity of CVE to fail a build. Use with the **--fail** flag. For example, **contrast audit --fail --severity high**. Severity levels are *critical*, *high*, *medium*, *low*, or *note*.

- **--track**

By default, results are not held or stored, which would allow you to do local checks via your console. Add the **--track** flag to view your projects' SCA results under the **Static** view on the [Libraries \(page 924\)](#) page in the Contrast web interface. Note that this is only available for Contrast CLI V2.0.

- **Advanced options:**

- **--api-key**

Required for Enterprise users. Agent API key provided by Contrast. See [agent keys \(page 104\)](#) to find your keys.

- **--application-id**

The ID of the application cataloged by Contrast.

- **--application-name**

The name of the application cataloged by Contrast.

- **--app-groups**

Assign your application to one or more preexisting groups when onboarding an application. Group lists should be comma separated.

- **--authorization**

Required for Enterprise users. Authorization header provided by Contrast.

- **--code**

The application code the application should use in Contrast.

- **--host**

Required for Enterprise users. The host name. For example, <https://app.contrastsecurity.com>.

- **--maven-settings-path**

Displays the path to the maven *settings.xml* file.

- **--metadata**

Define a set of key=value pairs (that conforms to RFC 2253) for specifying user-defined metadata associated with the application.

- **--organization-id**

Required for Enterprise users. The ID of your organization in Contrast. See [agent keys \(page 104\)](#) to find the ID.

- **--tags**

Apply labels to an application. Labels must be formatted as a comma-delimited list. For example, `label1,label2,label3`.

- **Proxy settings:**

- **--cacert**

Displays the path to the CaCert (certificate authority (CA) certificates) file.

- **--cert**

Displays the path to the Cert (certificate) file.

- **--cert-self-signed**

For Contrast on-premises (EOP) users with a local install, will bypass the SSL certificate and recognize a self-signed certificate.

- **--key**

Displays the path to the Certificate Key.

- **--proxy**

Allows for connection via a proxy server. If authentication is required, provide the username and password with the protocol, host, and port. For example, "http://username:password@<host>:<port>".

To use `audit` in pipelines for failing builds, see the [Contrast SCA Action](#).

## assess

Reports vulnerabilities found at run-time on a server using a Contrast agent.

- **Usage:** `contrast assess [option]`

Example:

```
contrastuser@usercsa-C02GD0LUMD6TTY ~ % contrast assess
Configuration file found at "user_path"
Session created.
CONTRAST-001 - [HIGH] Path Traversal from "RawQuery" QueryString
Parameter on
"/pathTraversal/os.Open/:source/:mode" pagePath Traversal from "RawQuery"
QueryString Parameter on "/pathTraversal/os.Open/:source/:mode" page
App: CLIAssessApplication
Source: GET
/pathTraversal/os.Open/:source/:mode?
input=.%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2Fetc%2Fpa
sswd
Location: /opt/homebrew/Cellar/go/1.19.1/libexec/src/os/file.go, line
316, in os.Open()
Dataflow: "../../../../../../../../../../../../etc/passwd"
Issue: Because there is untrusted data being used as part of the
file path, it may be possible
for an attacker to read sensitive data or write, update, or
delete arbitrary files on the
container's file system. The ability to write arbitrary files
to the file system is also
called Unrestricted or Arbitrary File Uploads.

CONTRAST-002 - [HIGH] Path Traversal from "RawQuery" QueryString
Parameter on
"/pathTraversal/os.ReadFile/:source/:mode" pagePath Traversal from
"RawQuery" QueryString Parameter on "/pathTraversal/
os.ReadFile/:source/:mode" page
App: CLIAssessApplication
Source: GET
/pathTraversal/os.ReadFile/:source/:mode?
input=.%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2Fetc%2Fpa
sswd
Location: /opt/homebrew/Cellar/go/1.19.1/libexec/src/os/file.go, line
672, in os.ReadFile()
Dataflow: "../../../../../../../../../../../../etc/passwd"
Issue: Because there is untrusted data being used as part of the
file path, it may be possible
for an attacker to read sensitive data or write, update, or
delete arbitrary files on the
container's file system. The ability to write arbitrary files
to the file system is also
called Unrestricted or Arbitrary File Uploads.

CONTRAST-003 - [HIGH] Path Traversal from "input[0]" Parameter on "/"
pathTraversal/os.Open/:source/:mode"
```

```

pagePath Traversal from "input[0]" Parameter on "/pathTraversal/
os.Open/:source/:mode" page
 App: CLIAssessApplication
 Source: POST /pathTraversal/os.Open/:source/:mode

input=..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2Fetc%2Fpa
sswd
 Location: /opt/homebrew/Cellar/go/1.19.1/libexec/src/os/file.go, line
316, in os.Open()
 Dataflow: "../../../../../../../../../../../../etc/passwd"
 Issue: Because there is untrusted data being used as part of the
file path, it may be possible
 for an attacker to read sensitive data or write, update, or
delete arbitrary files on the
 container's file system. The ability to write arbitrary files
to the file system is also
 called Unrestricted or Arbitrary File Uploads.

```

#### • Options:

- `--config-path <path>`  
Specifies the path or directory for the `contrast_security.yaml` file that the Assess CLI and the agent share.  
If not specified, the default paths are:
  - **MacOS and Linux:** `/etc/contrast`
  - **Windows:** `%ProgramData%\Contrast\`
 Alias: `-c`
- `--file <filename>`  
Specifies the path or directory for the vulnerability results file so Contrast can read it and display the results in the terminal. The file name is `contrast-assess-{Date}.jsonl.`, where the date is in epoch milli-seconds, For example: `contrast-assess-1691520302714.jsonl.`  
Alias: `-f`
- `--help`  
Displays usage information for all `assess` command options.
- `--no-watch [true|false]`  
If set to `true` when using Assess with a Contrast agent, the CLI does not watch (or poll) Contrast for available vulnerabilities. The CLI retrieves the vulnerabilities only once for a specific `buildNumber`. The default setting is `false`.  
Alias: `-n`
- `--output-path <path>`  
Specifies the path or directory where you want the vulnerability results file located. The output file is in JSONL format. The file name is `contrast-assess-{Date}.jsonl.`, where the date is in epoch milli-seconds, For example: `contrast-assess-1691520302714.jsonl.`  
Alias: `-o`
- `--report-notes [true|false]`  
If set to `true`, the access command displays vulnerabilities with a notes severity level. The default value is `false` which displays higher priority vulnerabilities.  
Alias: `-r`

## sarif

Generates a SARIF file (`contrast.sarif`) that contains findings from Contrast Assess and Contrast SCA for a specific application ID.

**NOTE**

You can also use the `sarif` API to generate the SARIF file.

- **Usage:** `contrast sarif [option]`

Example

```
contrast sarif
--application-id 8f32952c-987c-4b9e-882c-a2b59a2fb4ee
--severity high
--metadata
'repo=TS,commit=commit-49-61670e50-1e9c-11ef-9109-059d9bfadadd,developer=D
ev-49-61670e50-1e9c-11ef-9109-059d9bfadadd'
```

- **Options:**

- `--application-id <id>`  
The ID of the application cataloged by Contrast.
- `--severity [type]`  
Set the severity level to filter the findings included in the SARIF output. Severity levels are **critical**, **high**, **medium**, **low** or, **note**.  
The severity level you specify is the minimum level that the report includes. For example, if you specify `--severity high`, the report includes findings with a severity of **high** and **critical**.
- `--metadata`  
Define a set of key=value pairs (that conforms to RFC 2253) for specifying user-defined metadata associated with the application.
- `--tool-type`  
Filters the type of findings included in the report. The valid types are **SCA** and **ASSESS**. For example, `--tool-type ASSESS` includes Assess findings only.  
If you don't specify this option, the report includes findings for both SCA and Assess.

- **Advanced options:**

- `--api-key`  
Required for Enterprise users. Agent API key provided by Contrast. See [agent keys \(page 104\)](#) to find your keys.
- `--authorization`  
Required for Enterprise users. Authorization header provided by Contrast.
- `--code`  
The application code the application should use in Contrast.
- `--host`  
Required for Enterprise users. The host name. For example, `https://app.contrastsecurity.com`.
- `--organization-id`  
Required for Enterprise users. The ID of your organization in Contrast. See [agent keys \(page 104\)](#) to find the ID.

- **Proxy settings:**

- `--cacert`  
Displays the path to the CaCert (certificate authority (CA) certificates) file.
- `--cert`  
Displays the path to the Cert (certificate) file.
- `--cert-self-signed`  
For Contrast on-premises (EOP) users with a local install, will bypass the SSL certificate and recognize a self-signed certificate.

- `--key`  
Displays the path to the Certificate Key.
- `--proxy`  
Allows for connection via a proxy server. If authentication is required, provide the username and password with the protocol, host, and port. For example, "http://username:password@<host>:<port>".

## scan

Performs a security SAST scan.

- **Usage:** `contrast scan [option]`

Example:

```
contrastuser@usercsa-C02GD0LUMD6TTY ~ % contrast scan
Searching for files to scan from from /Users/contrast/Documents/
Searched 3 directory levels & found...
- spring-petclinic-1.5.1.jar
- webgoat-server-8.2.2.jar
- webgoat.jar

Java Scan requires a .war or .jar file. Javascript Scan requires a .js
or .zip file.
To start a Scan enter "contrast scan -f <path-to-file>"
contrastuser@usercsa-C02GD0LUMD6TTY ~ % contrast scan -f webgoat.jar
Found existing project...
Uploading...
Uploaded file successfully.
Contrast Scan started.
```

### Here are your top priorities to fix

```
CRITICAL sql-injection (2)
 1. org/owasp/webgoat/plugin/challenge6/Assigment6.java @43
 2. org/owasp/webgoat/plugin/challenge5/challenge6/
Assigment5.java @38
```

- `--branch`  
Specifies a branch in a repository to be scanned. When specified, scan results are aggregated against results for the current branch and not the main project.  
Alias: `-b`
- `--fail`  
Fail a build based on the severity of the vulnerability found. Use with the `--severity` flag. For example, **`contrast scan --fail --severity high`**. Returns all failures if no severity level is specified. If a failure is detected the CLI will exit with code 2.
- `--file`  
Path of the file you want to scan. Contrast searches for a .jar, .war, .js or .zip file in the working directory if a file is not specified.  
Alias: `-f`
- `--help`  
Displays usage information for all `scan` command options.
- `--host`  
Required for Enterprise users. The host name. For example, `https://app.contrastsecurity.com`.
- `--language`  
Valid values are JAVA and JAVASCRIPT.

**IMPORTANT**

This option is not valid if you are using the multi-language source code scan engine.

Alias: -l

- --memory

Memory override for the multi-language source code scan engine. The default memory setting is 2 GB.

- --name

Contrast project name. If not specified, Contrast uses *contrast.settings* to identify the project or creates a project.

Alias: -n

- -r

Contrast resource group name. This option is required for host-based customers if role-based access control is turned on.

- --save

Download the results to a Static Analysis Results Interchange Format (SARIF) file. The file is downloaded to the current working directory with a default name of *results.sarif*. You can view the file with any text editor.

Alias: -s

- --severity

A Contrast vulnerability severity level that returns a build failure status code that you can use to gate builds in pipelines.

Valid values are: *critical*, *high*, *medium*, *low*, and *note*.

The specified value is the minimum level of severity that returns a build failure status code. For example, if you specify `--severity high`, a finding of that severity or higher returns a build failure status code.

Use with the `--fail` flag. For example, **`contrast scan --fail --severity high`**.

- --timeout

Time in seconds to wait for the scan to complete. Default value is 300 seconds.

Alias: -t

- **Options:**

- **Advanced options:**

- --api-key

Required for Enterprise users. Agent API key provided by Contrast. See [agent keys \(page 104\)](#) to find your keys

- --authorization

Required for Enterprise users. Authorization header provided by Contrast.

- --ff

Fire and forget. Do not wait for the result of the scan.

- --host

Required for Enterprise users. The host name. For example, `https://app.contrastsecurity.com`.

- --label

Adds a label to the scan. Defaults to *Started by CLI tool at [current date]*.

- --organization-id

Required for Enterprise users. The ID of your organization in Contrast. See [agent keys \(page 104\)](#) to find the ID.

- --project-id

The ID associated with a scan project. To find the ID, select a scan project in Contrast and locate the last number in the URL.

- **Proxy settings:**

- `--cacert`  
Displays the path to the CaCert (certificate authority (CA) certificates) file.
- `--cert`  
Displays the path to the Cert (certificate) file.
- `--cert-self-signed`  
For Contrast on-premises (EOP) users with a local install, will bypass the SSL certificate and recognize a self-signed certificate.
- `--key`  
Displays the path to the Certificate Key.
- `--proxy`  
Allows for connection via a proxy server. If authentication is required, provide the username and password with the protocol, host, and port. For example, "http://username:password@<host>:<port>".

## lambda

Name of AWS lambda function to scan.

- **Usage:** `contrast lambda --function-name <function> [options]`
- **Alias:** `-f`
- **Options:**
  - `--endpoint-url`  
AWS Endpoint override. Similar to AWS CLI.  
Alias: `-e`
  - `--help`  
Displays usage information for all `lambda` command options.
  - `--region`  
Region override. Defaults to `AWS_DEFAULT_REGION`. Similar to AWS CLI.  
Alias: `-r`
  - `--profile`  
AWS configuration profile override. Similar to AWS CLI.  
Alias: `-p`
  - `--json`  
Return response in JSON (versus default human-readable format).  
Alias: `-j`
  - `--verbose`  
Returns extended information to the terminal.  
Alias: `-v`
  - `--list-functions`  
Lists all available lambda functions to scan.
  - `--help`  
Displays usage guide.  
Alias: `-h`
- **Proxy settings:**
  - `--cacert`  
Displays the path to the CaCert (certificate authority (CA) certificates) file.
  - `--cert`  
Displays the path to the Cert (certificate) file.
  - `--cert-self-signed`  
For Contrast on-premises (EOP) users with a local install, will bypass the SSL certificate and recognize a self-signed certificate.

- `--key`  
Displays the path to the Certificate Key.
- `--proxy`  
Allows for connection via a proxy server. If authentication is required, provide the username and password with the protocol, host, and port. For example, "http://username:password@<host>:<port>".

## Help and learn

### help

Displays usage guide. To list detailed help for any CLI command, add the `-h` or `--help` flag to the command.

- **Usage:** `contrast help`

Example:

```
contrastuser@usercsa-C02GD0LUMD6TTY ~ % contrast help
Contrast CLI @ v1.0.24
Contrast Scan CLI
Pre-requisites Java, Javascript and .NET supported

To scan a Java project you will need a .jar or .war file for analysis

To scan a Javascript project you will need a single .js or a .zip of
multiple .js files

To scan a .NET c# webforms project you will need a .exe or a .zip file
for
analysis

The file argument is optional. If no file is given, Contrast will search
for
a .jar, .war, .exe or .zip file in the working directory.

Submitted files are encrypted during upload and deleted in 24 hours.

Scan Options
-l, --language string (optional): Valid values are JAVA, JAVASCRIPT and
DOTNET
--label string (optional): adds a label to the scan - defaults
to 'Started by CLI tool at
current date'

-n, --name string (optional): Contrast project name. If not
specified, Contrast uses
contrast.settings to identify the project or creates a project.

-f, --file string (optional): Path of the file you want to scan. If
no file is specified,
```



Contrast searches for a .jar, .war, .exe or .zip file in the working directory.

-t, --timeout number (optional): Time in seconds to wait for scan to complete. Default value is 300 seconds.

--fail (optional): Use with contrast scan or contrast audit. Detects failures based on the severity level specified with the --severity command. For example, "contrast scan --fail --severity high". Returns all failures if no severity level is specified.

--severity type (optional): Use with "contrast scan --fail --severity high" or "contrast audit --fail --severity high". Set the severity level to detect vulnerabilities or dependencies. Severity levels are critical, high, medium, low or note.

-s, --save string (optional): Saves the Scan Results SARIF to file. Advanced

-o, --organization-id string (required for Contrast Enterprise): The ID of your organization as provided

by Contrast UI

--api-key string (required for Contrast Enterprise): An agent API key as provided by Contrast UI

--authorization string (required for Contrast Enterprise): An authorization header as provided by Contrast UI

--host string (required for Contrast Enterprise): host name e.g. https://app.contrastsecurity.com

--proxy string (optional): Allows for connection via a proxy server. If authentication is required please provide the username and password with the protocol, host and port. For instance: "https://username:password@<host>:<port>".

--key string (optional): Path to the Certificate Key

--cacert string (optional): Path to the CaCert file

--cert string (optional): Path to the Cert file

--cert-self-signed (optional): For EOP users with a local Teamserver install, this will bypass the SSL certificate and recognise a self signed certificate.

```
-p, --project-id string (optional): The ID associated with a scan project. Replace <ProjectID> with the ID for the scan project. To find the ID, select a scan project in
```

Contrast and locate the last number in the URL.

```
-l, --language string (optional): Valid values are JAVA, JAVASCRIPT and DOTNET
--ff (optional): Fire and forget. Do not wait for the result of the scan.
--label string (optional): adds a label to the scan - defaults to 'Started by CLI tool at current date'
```

Need More Help? NEW users

Check out: <https://support.contrastsecurity.com>

Learn more at: <https://www.contrastsecurity.com/developer>

Join the discussion: <https://www.contrastsecurity.com/developer/community>

Existing Contrast Licensed user?

Read our docs: <https://docs.contrastsecurity.com/en/run-contrast-cli.html>

Want to UP your game? type 'contrast learn'

Advance your security knowledge and become an All-star coder with

Contrast Secure Code Learning Hub.

- **Alias:** -h

## learn

Launch [Contrast's Secure Code Learning Hub](#).

- **Usage:** contrast learn  
Example:

```
contrastuser@usercsa-C02GD0LUMD6TTY ~ % contrast learn
Opening Contrast's Secure Code Learning Hub...
If the page does not open you can open it directly via https://
www.contrastsecurity.com/developer/learn
```

## Legacy Contrast CLI



### IMPORTANT

Legacy Contrast CLI will be deprecated as of October 2022. We encourage you to begin using the [new Contrast CLI \(page 994\)](#).

Use the Contrast command line interface (CLI) to analyze libraries at the earliest stage of the software development life cycle (SDLC).

The Contrast CLI runs on Node.js but can be used on any application to provide composition analysis capabilities at the command line. For details about the supported platforms and languages, see the [Contrast CLI supported languages \(page 1015\)](#) page.

With this composition analysis you can:

- Identify vulnerable libraries
- Fail a build based on CVE severity
- View a [dependency tree \(page 934\)](#) to understand the dependencies between libraries and where vulnerabilities have been introduced
- Identify node.js libraries at risk for dependency confusion
- Generate SBOM

Contrast does this by supplementing existing runtime instrumentation from Contrast agents, with data from pre-compile analysis (typically not available at runtime).

Install the [Contrast CLI \(page 1015\)](#) so you can [register new applications \(page 1016\)](#) and begin analyzing your libraries during the development phase using the [command line options. \(page 1018\)](#)

## Legacy Contrast CLI - supported languages

We support the following languages for Contrast CLI:

Language	Requirements	Notes
Java	Maven	A Maven project must be defined with a <code>pom.xml</code> file, and have the <a href="#">Apache Maven Dependency plugin</a> . To test if the CLI works with your project, build a dependency tree by running <code>mvn dependency:tree</code> .
	Gradle (v4.8 or above)	A <code>build.gradle</code> file is required and <b>gradle dependencies</b> or <b>./gradlew dependencies</b> must also be supported.
.NET Framework and .NET Core	MSBuild 15.0 or greater and a <code>packages.lock.json</code> file	If the <code>packages.lock.json</code> file is not present, it can be generated by setting <code>RestorePackagesWithLockFile</code> to <b>true</b> within each <code>csproj</code> and running .NET build.
Node.js	You must have either a <code>package-lock.json</code> or a <code>yarn.lock</code> file present.	Vulnerability reporting is supported for front-end technologies like React or Angular.
PHP	You must have the <code>compose.lock</code> and the <code>composer.json</code> files present.	
Python	The <code>pipfile</code> and <code>pipfile.lock</code> files.	
Ruby	The <code>gemfile</code> and <code>gemfile.lock</code> files.	
Go	The <code>go.mod</code> file.	



### NOTE

Only single language applications are supported at this time.

## Legacy Contrast CLI - Install

To install the [Contrast CLI \(page 1014\)](#):

1. [Install Node.js](#). The Contrast CLI is executed as a Node.js package, so this is required. Versions 10, 12, and 14 are currently supported.
2. Instrument your application.

**NOTE**

It is also possible to [register an application \(page 1016\)](#) that has not yet been instrumented. However, all applications should be instrumented so that your application has a [library score \(page 934\)](#) and the data in the library grid is populated.

3. Use the `cli_proxy` property in your agent configuration to establish communication with Contrast over a proxy.  
If authentication is required, provide the username and password with the protocol, host and port.  
For example:

```
http://username:password@<host>:<port>
```

4. Be sure the source code for target applications is available locally. Follow the requirements for your application's [language \(page 1015\)](#).
5. Run the following command:

```
npm install -g @contrast/contrast-cli
```

Alternatively, you can install the CLI with Yarn with the following command:

```
yarn global add @contrast/contrast-cli
```

**NOTE**

The Contrast CLI must be installed globally.

6. Once the installation is complete you can [register an application \(page 1016\)](#) to begin analyzing your code.

## Legacy Contrast CLI - Register applications

Once you [install the Contrast CLI \(page 1015\)](#) you must first register applications in order to see the results in Contrast.

**TIP**

You may want to invoke the Contrast CLI as part of your automated build process.

1. Locate your application ID. The application ID is the last URI segment in the Contrast URL in your browser.



2. Locate your [keys \(page 598\)](#). You will need:
  - API key
  - Organization ID
  - Authorization header
  - Server host name from the Contrast URL

**NOTE**

You only need to enter the server host name. For example, if the Contrast URL is `https://app.contrastsecurity.com/file/path/`, just enter:

```
--host app.contrastsecurity.com
```

3. To begin analysis, use one of these options:

- Replace `<APIKey>`, `<AuthorizationKey>`, `<OrganizationID>`, `<Host>` and `<ApplicationID>` with your API key, authorization header, Organization ID, host name and application ID, then run the CLI.

```
contrast-cli \
--api_key <APIKey> \
--authorization <AuthorizationKey> \
--organization_id <OrganizationId> \
--host <Host> \
--application_id <ApplicationId>
```

- Place credentials within a YAML file, using the same replacements:

```
cli:
 api_key: <APIKey>
 authorization: <AuthorizationKey>
 organization_id: <OrganizationId>
 host: <Host>
 application_id: <ApplicationId>
```

Replace `<path/to/yaml>` with your YAML path, and run this command to initiate:

```
contrast-cli --yaml_path <path/to/yaml>
```

**NOTE**

If you need to go through a communication protocol like Transport Layer Security (TLS) for example add the following parameters to the YAML file:

```
key: pathToKey
cert: pathToCert
cacert: pathToCaCert
```

4. After you see a success message, you are ready to [view the dependency tree \(page 934\)](#).

**TIP**

It is possible to add a new application to Contrast without instrumenting the application by using the `--catalogue_application` and `--application_name` options. However, it is best to [instrument the application \(page 52\)](#) so that the [library score \(page 934\)](#) and library grid are populated in Contrast.

For example:

```
contrast-cli \
--catalogue_application \
--api_key <YourApiKey> \
--authorization <YourAuthorizationKey> \
--organization_id <YourOrganizationID> \
--host <YourHost> \
--application_name <YourApplicationName> \
--language <YourApplicationLanguage>
```

Replace `<APIKey>` with your API key, `<AuthorizationKey>` with the authorization header, `<OrganizationID>` with your organization ID, `<Host>` with your host name, `<ApplicationName>` with your application name, and `<ApplicationLanguage>` with your application language. Allowable language values are JAVA, DOTNET, NODE, PHP, PYTHON, RUBY, and GO.

You will know the catalogue operation was successful if an application ID is displayed in the console.

**NOTE**

You can also register an application and create an SBOM report at the same time with [a set of CLI commands \(page 1018\)](#).

## Legacy Contrast CLI - commands

The CLI offers a command line help guide with the `-h` or `--help` option. The help guide contains the following commands to help you understand more about Contrast configuration, applications, and vulnerabilities.

In the following examples, replace `<string>` or `<level>` with the string or level value that applies to your particular situation.

### General commands

Commands for connection and configuration.

Command	Description
<code>--api_key &lt;string&gt;</code>	An agent <a href="#">API key (page 104)</a> provided by Contrast. (required)
<code>--application_id &lt;string&gt;</code>	The ID of the application cataloged by Contrast. (required)
<code>--application_name &lt;string&gt;</code>	The name of the application cataloged by Contrast. (optional)

Command	Description
<code>--authorization &lt;string&gt;</code>	User authorization credentials provided by Contrast. (required)
<code>-h, --help</code>	Displays the help guide.
<code>--host &lt;string&gt;</code>	The name of the host and, optionally, the port expressed as <code>&lt;host&gt;:&lt;port&gt;</code> . Does not include the protocol section of the URL ( <i>https://</i> ). Defaults to <i>app.contrastsecurity.com</i> . (optional)
<code>--language &lt;string&gt;</code>	Valid values are JAVA, DOTNET, NODE, PHP, PYTHON, RUBY, and GO. If there are multiple project configuration files in the <code>project_path</code> , language is required. (required for catalogue)
<code>--organization_id &lt;string&gt;</code>	The ID of your organization (page 104) in Contrast. (required)
<code>--project_path &lt;string&gt;</code>	The directory root of a project/application that you want to analyze. Defaults to the current directory. (optional; required if running on Windows).
<code>--proxy &lt;string&gt;</code>	Allows for connection over a proxy server. If authentication is required, provide the username and password with the protocol, host and port. For example, <i>http://username:password@&lt;host&gt;:&lt;port&gt;</i> . (optional)
<code>--silent</code>	Silences JSON output. (optional)
<code>--sub_project &lt;string&gt;</code>	Specifies the subproject within a Gradle application. (optional)
<code>-v, --version</code>	Displays the CLI version you are currently using.
<code>--yaml_path &lt;string&gt;</code>	<p>The path to display parameters from the YAML file (optional)</p> <p>If <code>yaml_path</code> is used, the following connection parameters are ignored from the terminal:</p> <ul style="list-style-type: none"> <li><code>yamlOnly:</code></li> <li><code>key:pathToKey</code></li> <li><code>cert:pathToCert</code></li> <li><code>cacert:pathToCaCert</code></li> </ul>



## NOTE

Parameters in these commands may need to be quoted to avoid issues with special characters. For example:

```
--application_name = "My_app_name_${+}=(/\\"
```

## SCA

Commands related to Contrast SCA examination.

Command	Description
<b>Catalog applications</b>	
<code>--app_groups &lt;string&gt;</code>	Assigns your application to one or more pre-existing groups when using the <code>catalogue</code> command. Group lists should be comma separated. (optional)
<code>--catalogue_application</code>	Catalog an application (required). If the application name does not exist, create the application and send the dependency tree, else append the dependency tree to an existing application.
<code>--code &lt;string&gt;</code>	The application code this application should use in Contrast. (optional)

Command	Description
--metadata <string>	Define a set of key=value pairs (which conforms to RFC 2253) for specifying user-defined metadata associated with the application. (optional)
--tags <string>	Apply labels to an application. Labels must be formatted as a comma-delimited list. Example - label1,label2,label3 (optional)
Snapshot - default command does not have a command on the terminal. Java only.	
--maven_settings_path <PathToFile>	Allows you to specify an alternative location for your maven settings.xml file. Replace <PathToFile> with the full path for the file. Add this path to the full set of keys when you register your application with the CLI (page 1016). (optional)
Register an application	
--cli_api_key <string>	Use this set of commands (values described in the tables above and below) to register an application and get an SBOM report at the same time.
--cli_authorization <string>	
--cli_organization_id <string>	Note: The "cli_" prefix in the parameters will be deprecated in a future release.
--cli_host <string>	
--language <string>	
--application_name <string>	
--sbom	
Reports	
--cve_severity <level>	Combined with --report, allows the user to report libraries with vulnerabilities above a chosen severity level (page 1039). For example, cve_severity medium only reports vulnerabilities at Medium or higher severity.
--cve_threshold <number>	Sets the number of CVEs allowed before a build is failed. If there are more CVEs than the threshold, the build will fail.
--fail	Fails the build if any vulnerabilities are found. Can be used in combination with cve_severity to fail builds with vulnerabilities at severity levels defined by the user.
--report	Shows a report of vulnerabilities in the application from compile time.
--ignore_dev	Combined with the --report command excludes developer dependencies from the vulnerabilities report. By default, all dependencies are included in a report.
SBOM	
--sbom	Generate and download a Software Bill of Materials (SBOM) (page 1046) in CycloneDX JSON format





### TIP

The `--report` command can be used to return details of all vulnerable libraries in the terminal response. Every CVE found will have output like this:

```
org.webjars/jquery-ui/1.11.4 is vulnerable
```

```
CVE-2016-7103 MEDIUM Cross-site scripting (XSS) vulnerability
in jQuery UI before 1.12.0 might allow remote attackers to
inject arbitrary web script or HTML via the closeText parameter
of the dialog function.
```

The vulnerable records returned can be restricted by using the `--cve_severity` parameter which sets the minimum threshold for a CVE to be reported.

To prevent an application from being deployed with a library above a severity threshold the `--fail` parameter can be used as part of an automated CI/CD pipeline. For example, you can run the CLI using a YAML file with:

```
contrast-cli --yaml_path path/to/yaml --report --cve_severity
high --fail
```

## Scan

Commands related to Contrast Scan. See also [Integrate scans with builds \(page 907\)](#).

Contrast Scan supports EXE and ZIP files for .NET projects. The language must be set to DOTNET and the ZIP should be a ZIP of the `./bin` folder that contains your dlls.

Command	Description
<code>--project_id &lt;ProjectID&gt;</code>	<p>The ID associated with a scan project. Replace <code>&lt;ProjectID&gt;</code> with the ID for the scan project. To find the ID, select a scan project in Contrast and locate the last number in the URL.</p> <p><b>Recommended:</b> For the first scan, use the <code>--project_name</code> command instead of this one. Scan creates the project for you.</p>
<code>--project_name &lt;ProjectName&gt;</code>	<p>The name of the scan project. If the name includes spaces, enclose it in double quotes ("").</p> <p>If you specify a new name, Scan creates the project. If you specify the name of an existing project, Scan adds the uploaded file to that project.</p>
<code>--save_scan_results</code>	If provided, will save the SARIF file as <code>results.json</code> to the current directory. (optional)
<code>--scan&lt;FileToBeScanned&gt;</code>	Starts a static scan of the specified WAR or JAR file. Replace <code>&lt;FileToBeScanned&gt;</code> with the path of the WAR or JAR file that you want to upload for scanning.
<code>--scan_results_file_name</code>	Must be JSON file format. If provided, will override the default name to save the SARIF file. (optional)
<code>--scan_timeout</code>	Set a specific time span (in seconds) before the function times out. The default timeout is 20 seconds if <code>scan_timeout</code> is not set.
<code>--wait_for_scan</code>	Waits for the result of the scan.

## Vulnerabilities

Once you [instrument an application \(page 52\)](#), Contrast shows you all the vulnerabilities it's discovered, addressing some of the [most common vulnerabilities](#) and many others.

Contrast agents discover any code flaws that are in your applications, and report them. Contrast then presents and classifies these vulnerabilities with a severity level to help you prioritize and mark the vulnerabilities as needed.

- [View vulnerabilities \(page 1022\)](#)
- [View application vulnerabilities \(page 1024\)](#)
- [Analyze vulnerabilities \(page 1032\)](#)
- [Track vulnerabilities \(page 1029\)](#)
- [Fix vulnerabilities \(page 1033\)](#)

## Use Contrast AI guidance ☺

Contrast AI provides additional information about how to fix vulnerabilities that the Contrast IAST (Assess) technology discovers. The guidance it generates is specific to the frameworks and libraries that your application uses.

Contrast AI collects information about your frameworks and libraries from existing data stored in Contrast. It uses this information along with Anthropic to generate customized guidance for fixing a selected vulnerability.



### REMINDER

Use of Intelligent Remediation Guidance (the "Feature") means that you agree to submit data to Anthropic on AWS Bedrock that will be analyzed and used as authorized under the [Anthropic on Bedrock Terms of Service](#). Use of the Feature is entirely at your own risk.

## Before you begin

- Ensure that the [Contrast AI setting \(page 1160\)](#) is turned on at the organization level.
- This feature is available for hosted customers in the USA only.
- This feature is currently supported for vulnerabilities that Contrast IAST (Assess) technology finds.
- Contrast caches responses from the large language model (LLM) for 24 hours.

## Steps

To find Contrast AI details for vulnerabilities that the Contrast IAST technology finds:

1. In the header, select **Vulnerabilities**.
2. Select a vulnerability in the list.
3. Select the **How-to-Fix** tab.
4. Select **Use Contrast AI**.

Contrast AI generates supplementary information based on your application's technology.

## View vulnerabilities at an organization level

### Before you begin

- Exercise (browse or use) your application so Contrast can find weaknesses and present results in the Contrast application.

- To see your application's vulnerability data in more detail, configure your Contrast agent to report [session metadata](#) (page 616).
- To view the Protected in <environment> column, Protect must be turned on and you need these permissions:
  - If you are using role-based access control, you need a role with the View application and Access Protect actions.
  - If you are using organization users and groups, a SuperAdmin must turn on Protect for the user.

## Steps

1. In the header, select **Vulnerabilities**.
2. To display vulnerabilities for licensed applications only, select **Show licensed only** at the top of the vulnerabilities list.
3. To filter by columns, select the **Filter** icon (▼) next to the column headers. These filter options are available if applicable to the selected application:
  - **Severity:** Available filters are: Critical, High, Medium, Low, and Note.
  - **Vulnerability:** Available filters are:
    - **Vulnerability tags::** Vulnerabilities associated with custom tags that you created.
    - **Type:** Types of vulnerabilities.
    - **Servers:** Vulnerabilities for applications associated with selected servers.
    - **Environments:** Vulnerabilities for applications in selected environments: Development, QA, and production.
    - **Sinks:** Vulnerabilities that originate from a common sink.  
A sink is common custom code shared between multiple data-flow vulnerabilities.  
Filtering by sink can help you identify a line of code that is causing multiple vulnerabilities.
    - **URLs:** Vulnerabilities associated with a specific URL.
    - **Compliance policy:** Vulnerabilities associated with a compliance policy
  - **Protected in <environment>:** Available filters are:
    - **Off:** Protection is turned off.
    - **Monitor:** The protection rule that is mapped to the vulnerability is in Monitor mode.
    - **Monitor at perimeter:** The protection rule that is mapped to the vulnerability is in Monitor at perimeter mode.
    - **Block:** The protection rule that is mapped to the vulnerability is in Block mode.
    - **Block at perimeter:** The protection rule that is mapped to the vulnerability is in Block at perimeter mode.
    - **No rule available:** Contrast does not have a protection rule for the vulnerability.
  - **Application:** Available filters are:
    - **Application names:** Names associated with applications.
    - **Custom tags:** Tags assigned to applications.
    - **Languages:** The languages used for applications.
    - **Technologies:** Technologies that applications use. For example, JSON or jQuery.
    - **Application importance:** The importance level you set in the application settings.
    - **Application metadata:** Application metadata associated with applications.
  - **Last detected:** Available filters are: First or Last detected and Time range. Select **Custom** to enter specific dates and times.
  - **Status:** Available filters are Status and whether Contrast is tracking the vulnerability

To remove filters, select **Clear** next to the column header.



- To view vulnerability details, select a name. You can view details for these categories:
  - HTTP information
  - Steps on how to fix this vulnerability
  - Details about the identity, timing and location of the vulnerability including build numbers, reporting servers, category and security standards
  - Use the **Activity** tab to add new comments to the associated Jira issue (if it is [configured \(page 1098\)](#)) and, the other way around, if a comment is added to the Jira issue it will appear under the Activity tab.

## See also

[View application vulnerabilities \(page 1024\)](#)

## View application vulnerabilities

From the Applications list, you can view vulnerabilities for a specific application.

### Before you begin

- Exercise (browse or use) your application so Contrast can find weaknesses and present results in the Contrast application.
- To see your application's vulnerability data in more detail, configure your Contrast agent to report [session metadata \(page 616\)](#).
- To view the Protected in <environment> column, Protect must be turned on and you need these permissions:
  - If you are using role-based access control, you need a role with the View application and Access Protect actions.
  - If you are using organization users and groups, a SuperAdmin must turn on Protect for the user.

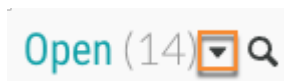
### Steps

1. Select **Applications** in the header.  
The Applications list displays the number of open vulnerabilities for each application. To view details for specific types of vulnerabilities (for example, critical or high), in the Open Vulnerabilities column, select the relevant section of the bar.



An open vulnerability has a status of Reported, Suspicious, or Confirmed.

2. Alternatively, In the Applications list, select an application name and then, select the **Vulnerabilities** tab. You see a list of vulnerabilities for that application.
3. In the Vulnerabilities tab, to filter vulnerabilities, select the small triangle at the very top of the list.



These filter options are available:

- Open
  - High confidence
  - Policy violation
  - Pending review
4. To search for specific vulnerabilities, select the magnifying glass icon (🔍).
  5. To view a timeline of the vulnerabilities, select the **trend line** symbol (📈) above the list .

Use the buttons above the chart to view data by **Severity** or **Discovery**. Hover over the trend lines to see a breakdown of the data for that point in time (number of vulnerabilities, time stamp, or status).

Any filters you apply in the list also update the data in the chart. Use the filter for the **Last detected** column to update the time span shown in the timeline.

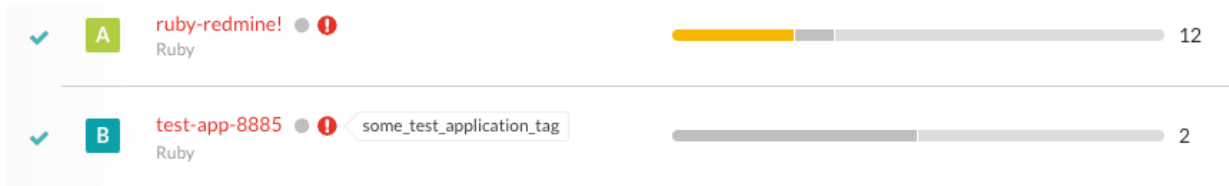
6. To filter by columns, select the Filter icon ▼ next to the column headers. These filters are available, if applicable to the selected application:
  - **Severity:** Available filters are: Critical, High, Medium, Low, and Note.
  - **Vulnerability:** Available filters, if applicable to the selected application, are:
    - **Vulnerability tags :** Custom tags you assigned to vulnerabilities
    - **Bugtrackers:** Whether you are using bugtracking integrations to track vulnerabilities.
    - **Type:** Types of vulnerabilities
    - **Modules:** Application modules associated with a vulnerability, including modules in a merged application.
    - **Servers:** Servers hosting the application.
    - **Environments:** Development, QA, and production
    - **Sinks:** Vulnerabilities that originate from a common sink  
A sink is common custom code shared between multiple data-flow vulnerabilities.  
Filtering by sink can help you identify a line of code that is causing multiple vulnerabilities.
    - **URLs:** Vulnerabilities associated with a specific URL.
    - **Compliance policy:** Vulnerabilities associated with selected compliance policies
    - **Routes:** Vulnerabilities associated with selected routes.
  - **Protected in <environment>:** Available filters are:
    - **Off:** Protection is turned off.
    - **Monitor:** The protection rule that is mapped to the vulnerability is in Monitor mode.
    - **Monitor at perimeter:** The protection rule that is mapped to the vulnerability is in Monitor at perimeter mode.
    - **Block:** The protection rule that is mapped to the vulnerability is in Block mode.
    - **Block at perimeter:** The protection rule that is mapped to the vulnerability is in Block at perimeter mode.
    - **No rule available:** Contrast does not have a protection rule for the vulnerability.
  - **Application:** Modules included in a merged application.  
If you are viewing vulnerabilities for a merged application, this filter shows view vulnerabilities for the modules in the merged application. This column is not displayed for applications that aren't merged.
  - **Last detected:** Available filters are: First or Last detected and Time range. Select **Custom** to enter specific dates and times.
  - **Status:** Available filters are Status and whether Contrast is tracking the vulnerability.
  - **Session:** This column is visible if you configured session metadata in an agent configuration file but haven't selected a session metadata filter. Use Session column filter to refine the results.  
Use the **View by** menu at the top of the list to filter the data by the session metadata values that you included in your agent configuration file. This filter updates the values shown in the Session column.  
The View by menu is visible if you configured session metadata in an agent configuration file but haven't selected a session metadata filter.

## Open vulnerabilities for merged applications

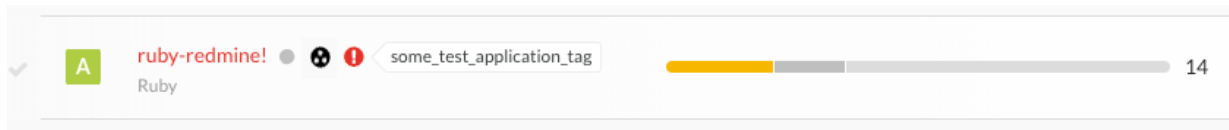
For merged applications, the Open Vulnerabilities column in the Applications list displays the number of vulnerabilities for all application modules in the primary application. The Applications list displays the primary application but not the modules in the primary application.

## Example:

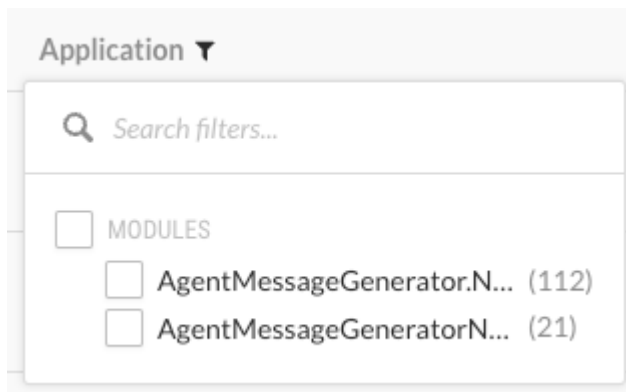
Before you merge applications, the Open Vulnerabilities column looks similar to this:



After you merge applications, the bar in the Open Vulnerabilities column shows vulnerabilities for the primary application and all the merged modules.



When you view the vulnerabilities tab for a merged application, use the filter for the Application column to view the modules where Contrast found the vulnerability.



## See also

[View vulnerabilities at an organization level \(page 1022\)](#)

## View vulnerability rates

The chart for open and closed vulnerability rates shows trends for the vulnerabilities for all applications in your organization.

## Steps

1. Display the new dashboard by selecting **View your new dashboard**.

**View your new dashboard (BETA)**

2. In the new dashboard, find the chart for **Open and closed vulnerability rates**.
3. Select a time frame for the chart:
  - **Last 7 days:** Displays vulnerabilities rates for the seven days before the current date. The chart shows data points for each day.
  - **Last 30 days:** Displays vulnerability rates for the 30 days before the current date. The chart shows data points for each day.
  - **Last 12 months:** Displays vulnerability rates for the 12 months before the current date.

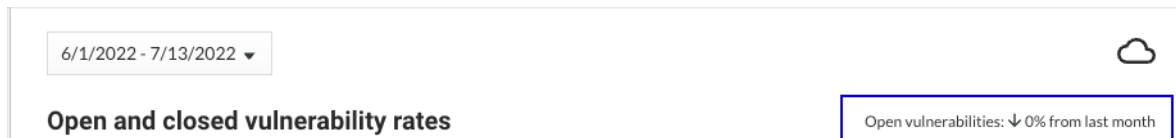
The chart shows data points for each month.

- **Custom:** Displays vulnerability rates for a selected time frame.

The chart shows data points for each day, each week, or each month, depending on the selected time frame.

If no data exists for part of a selected time frame, the chart displays no data for that part of the time frame. For example, if you selected a time frame of the Last 12 months but started using Contrast only nine months ago, the chart displays no data for the first three months of the time frame.

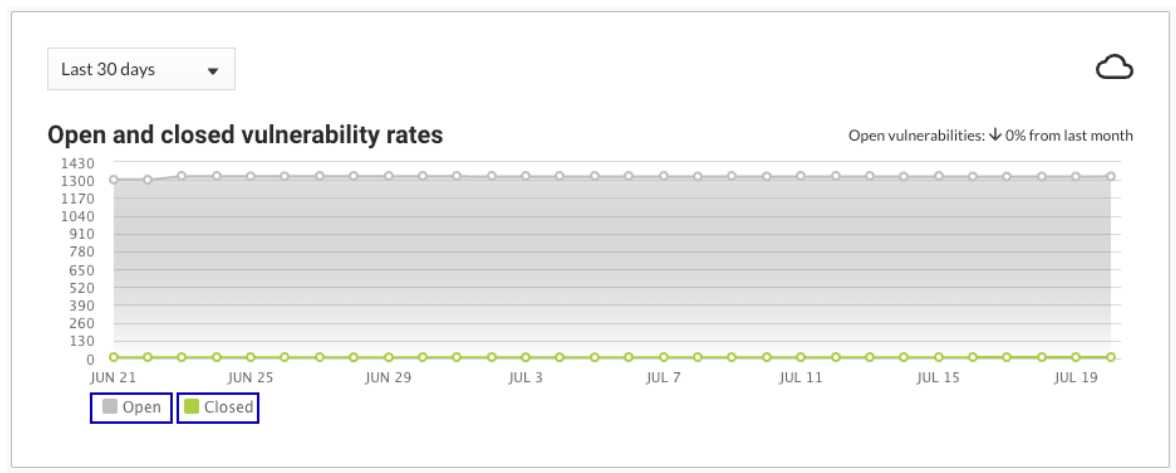
- To view the percentage of change during the selected time frame, look at the value at the top of the chart



- To filter the chart to display only open or only closed vulnerabilities, select the keys at the bottom of the chart.

Select the keys again to clear your selection.

- To view only open vulnerabilities, select the **Closed** key.
- To view only closed vulnerabilities, select the **Open** key.



## Add and delete vulnerabilities

When you instrument an application, vulnerabilities are automatically detected and they become **visible in Contrast** (page 1022). Depending on your particular security concerns, you can assess the risk of these vulnerabilities, eliminate false positives and prioritize fixes.

You may decide to delete a vulnerability if it is no longer useful.

To do this:

- Select **Vulnerabilities** in the header.
- Hover over the grid row with the vulnerability you want to delete and use the **Delete** icon in the right column. You can also find the icon in the top right of the vulnerability details page.  
To delete multiple vulnerabilities at once, use the check marks in the left column to select the vulnerabilities you want to delete, then select the **Delete** icon from the batch action bar that appears at the bottom of the page.



3. In the window that appears, select **Delete** to confirm your choice. Once confirmed, the vulnerability is removed and no longer appears in your list unless Contrast discovers it again.

## Group vulnerabilities by sink

Grouping vulnerabilities lets you combine vulnerabilities that share the same sink. The vulnerabilities in a group can affect multiple applications.

Grouping vulnerabilities reduces the number of entries that the vulnerability list shows. You can still see the data for individual vulnerabilities in a group.

## Before you begin

Group by sink applies only to vulnerabilities for applications with an Assess license. If you group by sink, **Show licensed only** is selected automatically.

## Steps

1. Select **Vulnerabilities** in the header.
2. At the top of the list, select **Group by sink**.



Depending on the filters you use for the list, you might need to scroll down to find the groups. A group looks similar to this example:

CRITICAL	2	SQL Injection sqlite3_adapter.rb, line 232, in execute()	Multiple	2 months ago	Reported
----------	---	-------------------------------------------------------------	----------	--------------	----------

The number indicates the number of vulnerabilities in the group.

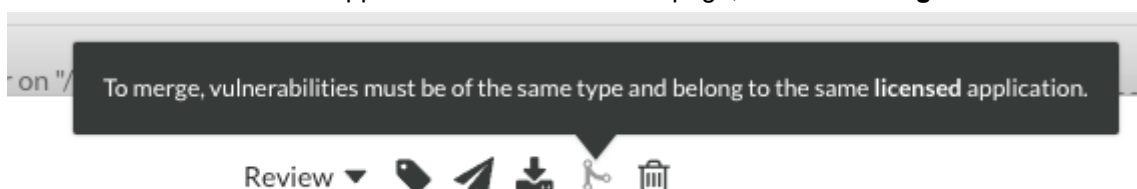
The values for the Severity, Application, and Status columns change to **Multiple**, if the vulnerabilities in a group have different severities, apply to different applications, or have different statuses.

3. To further refine the view, select one or more **Vulnerability** filters.
4. To view individual vulnerabilities in a group, select the group. Contrast shows a list of only the vulnerabilities in the group.
5. To remove all groups, clear the **Group by sink** checkbox.

## Merge vulnerabilities

If you find vulnerabilities of the same type from the same application, you can merge them to consolidate findings. To do this:

1. Select **Vulnerabilities** in the header.
2. Use the check marks in the left column to select two or more vulnerabilities you'd like to merge.
3. In the batch action bar that appears at the bottom of the page, select the **Merge** icon.



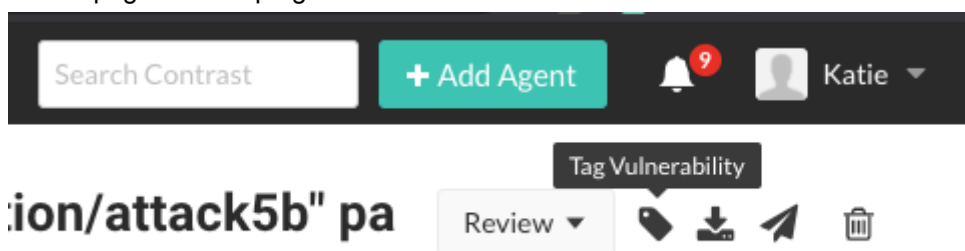
4. In the window that appears, select the vulnerability that you want to represent the merge.

## Add a tag to a vulnerability

You can tag vulnerabilities to better organize vulnerabilities and improve search in Contrast. To do this:

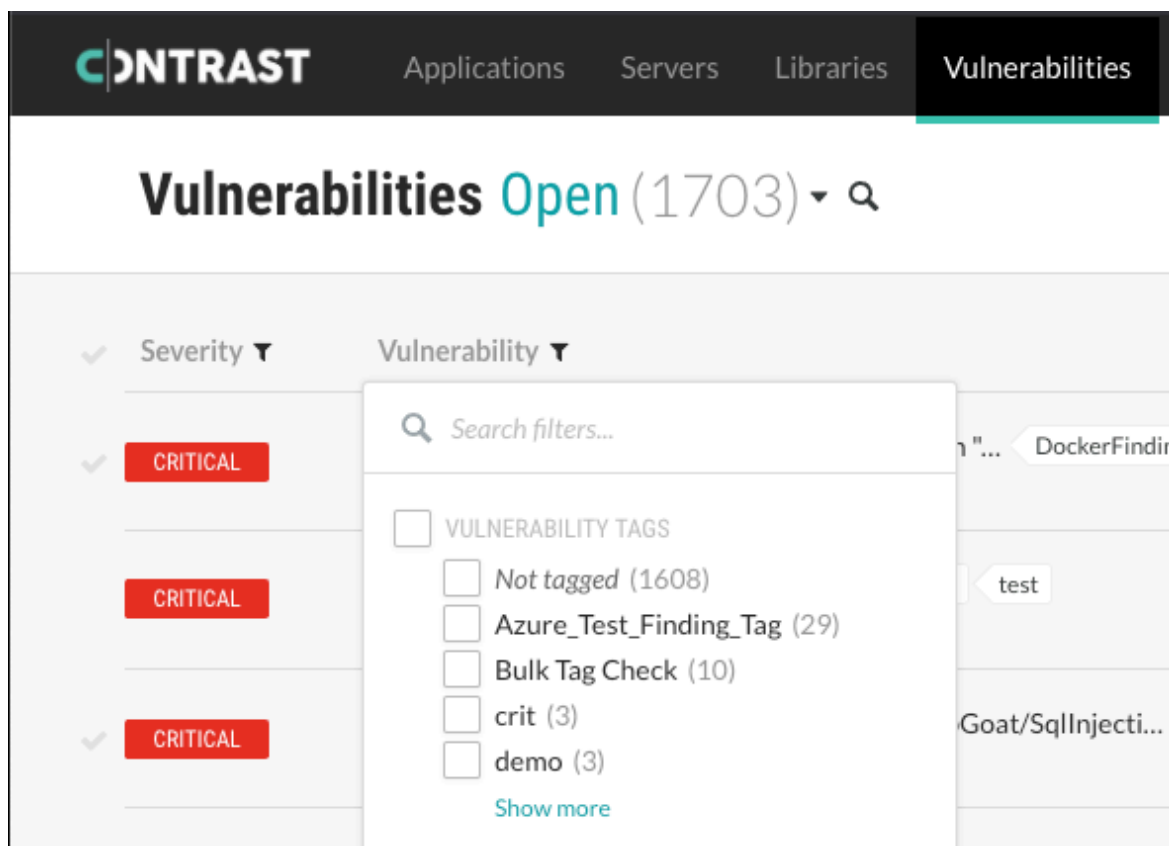


1. Select **Vulnerabilities** in the header, then hover over the row in the grid for the vulnerability you want to tag.
2. In the far right column, select the **Tag** icon. This option is also available from the vulnerability details page in the top right corner.



To tag multiple vulnerabilities, use the check marks in the left column of the vulnerabilities grid to select the ones you want to tag. In the batch action menu that appears at the bottom of the page, select the **Tag** icon.

3. In the window that appears, begin typing to see a list of tags. Select one or more from the dropdown, and/or type a new tag. To remove tags, select the **X**. Select **Save**.
4. To filter by tags, select the filter next to the **Vulnerability** column of the grid, then select the tags to filter.



5. You can also see tags next to the vulnerability name on the vulnerability's details page, and remove them by selecting the **X**.

## Track vulnerabilities

If you are using a bugtracker integration, you can track vulnerabilities in multiple ways:

- Send vulnerability data to other members of your organization.
- Plan and maintain timely patching to prevent attacks.
- Streamline workflows by sending vulnerability information directly to your bugtracking tool.

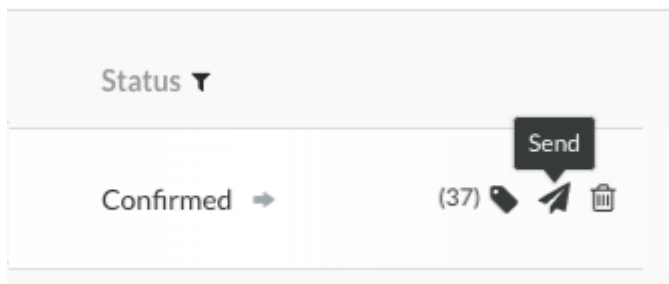
- Receive notifications of any new high or critical vulnerabilities in your application.

## Before you begin

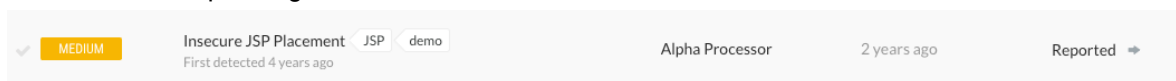
Be sure Contrast is integrated with at least one [bugtracker \(page 1051\)](#) tool.

## Steps

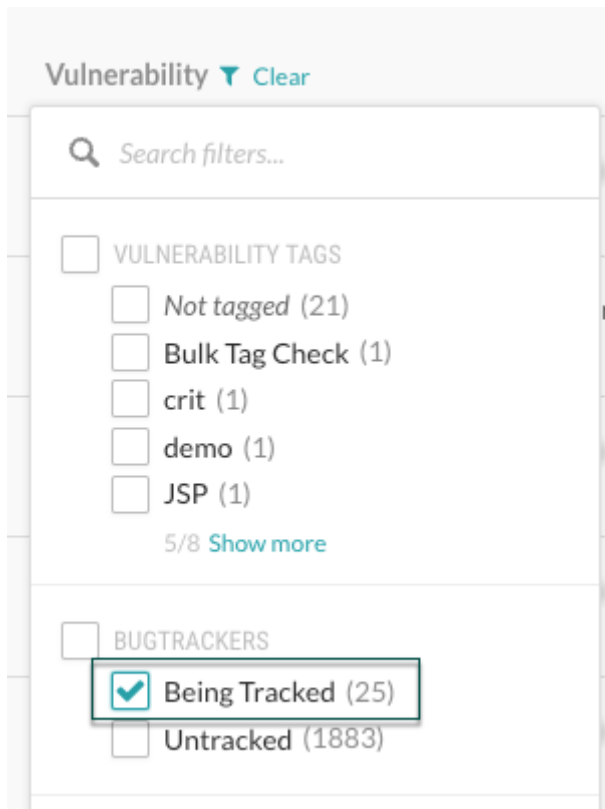
1. Select **Vulnerabilities** in the header.
2. To track a single vulnerability, hover over the end of the row for the vulnerability you want to track.
3. In the far right column, select the **Send** icon.  
This option is also available from the vulnerability details page.



4. To track multiple vulnerabilities:
  - a. Select the check marks next to each vulnerability you want to track.
  - b. In the batch action menu at the bottom of the page, select the **Send Vulnerability** icon (✈️).
  - c. Select **Send to bugtracker**.  
You can also choose to send the tracking data by email.
5. In the Send vulnerability window, select the bugtracker tool you want to use from the dropdown (if you are integrated with multiple tools), add any related information, and select **Send**.  
The vulnerability status updates automatically to **Reported** and an arrow icon displays next to the status of the vulnerability. Hover over the arrow for more information, including the bugtracker name and corresponding ticket numbers.



6. To quickly see which vulnerabilities are tracked, use the filter in the **Vulnerability** column and select **Being Tracked**.

**TIP**

You can also [export vulnerability data \(page 1033\)](#) to a CSV or XML file for custom processing, or use the API to gather data outside the web interface.

## Map Application and Detection Response (ADR) rules to Assess findings

Contrast can associate Assess findings with ADR rules. Doing so helps you make intelligent decisions about how to triage vulnerabilities.

This workflow illustrates how to get the best results from this association.

### Before you begin

- If role-based access control is turned on, you need a role with these actions: Access protect and View applications.
- If you are using organization users and groups, you need an Organization Admin role.
- If you select the Group by sink option on the Vulnerabilities list, no ADR rule status is displayed if a vulnerability group includes multiple applications, as different applications may be configured differently.

### Configure ADR rules

Configure the mode and environment for the ADR (Protect) rules that you want to use:

1. Under the user menu select **Policy management**.
2. Select **Protect rules**.


3. Select **Configure the default policy** at the top of the list.
4. [Change the mode for specific rules \(page 1133\)](#) to **Block** or **Monitor**.

## Configure rule mapping

Choose the environment to which the rule mode applies:

1. Under the user menu, select **Organization settings**.
2. Select **Applications**.
3. Under Map Protect rules to Assess finding, select an environment.  
The default setting is **Production**.  
Contrast applies the mode you configured for the mapped ADR rules to the selected environment.

## Determine actions to take

1. [Exercise your application \(page 117\)](#).  
As Contrast detects vulnerabilities, it displays them on the Vulnerabilities list. The Protected in environment column indicates the mode for the ADR rule mapped to each vulnerability. The column refers to the rule setting in the Contrast web interface.  
A case could exist where a specific agent is misconfigured to set Protect to Off. In this case, that server won't be protected until Protect is configured to On.
2. Take action:
  - You can change the mode for a rule to **Monitor** or **Block**.
    - To change the mode for a single rule, select the **Map** icon () at the end of the row for a vulnerability.
    - To change the mode for multiples rules at one time, [change the rule modes at an organization level \(page 1133\)](#).
  - You can choose to [lower the severity of a vulnerability \(page 896\)](#) because an ADR rule is protecting your application against that vulnerability.

## Analyze vulnerability events

Contrast provides information on what it observed when navigating your application by using vulnerability events. These events include the exact location where the vulnerability was found in the code and how the code was used. There are several types of events:

- **Source events:** Source events occur at the start of a scope. You can use the file and line number of the source event to see exactly where the call was made, and use the stacktrace in the source to understand how the program invoked the notable method. You can also view all the data related to the method, including the:
  - **Object:** The underlying object instance on which this call is invoked (if not a static call).
  - **Return:** The object returned from this call (or null, if void).
  - **Parameters:** The objects passed into this call.
- **Propagation events:** Each vulnerability may contain one or more propagation events. These events contain the same information as the source event, but they also have a type that indicates how the data was propagated. For example, a P2R propagation event takes the data from one or more of the parameters (the "P" in "P2R") and transfers it into the method return value (the "R" in "P2R").
- **Tag events:** These events add a tag, such as **validated** or **html-encoded**, to a vulnerability. These tags help eliminate false positives and provide clean, reliable results. They also contain the same contextual information as the other types of events. While tag events may occur within a vulnerability, they have nothing to do with the vulnerability discovered.
- **Trigger events:** The trigger is the last event in the vulnerability. The trigger is the call that makes the rule engine in the Contrast JVM Plugin perform its analysis, notice the vulnerability and generate the trace.



## IMPORTANT

Contrast only detects the actual behavior of an application. If a vulnerability doesn't represent a legitimate problem, an administrator [should update the applicable policy \(page 1120\)](#) to prevent this issue from occurring again. The most commonly reported false alarm is that the application has a custom control that Contrast doesn't know about.

On-premises users can add a custom method call to the appropriate tag list in the Contrast policy. For example, your custom HTML-encoder method that takes a string and returns an HTML-encoded string should add the **html-encoded** tag to the data.

You can use [Security controls \(page 1120\)](#) or [Application exclusions \(page 1142\)](#) to remediate false positives.

## Fix vulnerabilities

When a vulnerability arises, you need to assess the risk according to your particular security needs. If you decide to fix this vulnerability:

1. Learn more about the particular vulnerability by selecting the vulnerability name to open the details page. Then select the **How to fix** tab to see suggested steps to resolve the issue.
2. Fix the vulnerability as you see fit.
3. Check a fixed vulnerability. There are three ways to do this:
  - **Replay the request:** If the issue is remediated, you can replay the HTTP request. Select the **HTTP Info** tab to see if the issue is fixed. If it isn't fixed, the issue reappears with a status of Reported.
  - **Check build number:** For each application, you can assign a build version number. Use [session metadata \(page 616\)](#) to learn more about a vulnerability using the build number. Add this property to the `-javaagent` command:

```
-Dcontrast.override.appversion
```

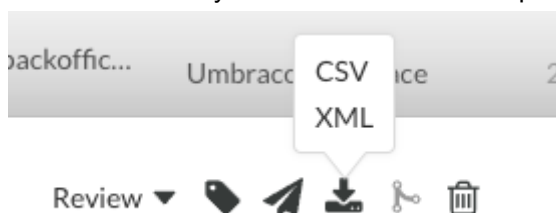
Provided you have set a build number during startup, you can use this as a filter and verify whether the issue still exists for this build version by clicking the Advanced link and the Build Number dropdown.

- **Check by time unit tests:** You can also filter by the time at which your unit tests were run, and set a date range to view your vulnerabilities in the Set Date Range input field above the vulnerabilities grid.


## Export vulnerability findings

To export vulnerability details:

1. Select **Vulnerabilities** in the header, then use the check marks in the left column of the vulnerabilities grid to select the vulnerability or vulnerabilities you want to use for the export.
2. In the batch action menu that appears at the bottom of the page, select the **Export** icon, then select the format you want to use for the export (CSV or XML).



Contrast begins to export the data.

3. Check the Notifications panel () to see when the export is completed. The notification contains a link for you to download the exported data.

Exports contain the following information for each vulnerability:

- Vulnerability Name
- Vulnerability ID
- Category
- Rule Name
- Severity
- Status
- Number of Events
- First Seen
- Last Seen
- Application Name
- Application ID
- Application Code
- CWE ID
- Request Method
- Request Port
- Request Protocol
- Request Version
- Request URI
- Request Qs
- Request Body
- Instance ID



### TIP

To create more complex custom software composition analysis reports about your applications, you can use the [Application API](#) to access Contrast vulnerability data.

You may also explore additional details on your vulnerabilities using a manual method.

For, example, this curl request retrieves a list of vulnerabilities that also shows a list of the applications in which each vulnerability was found. The jq tool formats the data as CSV for use in a custom report.

```
curl \
 -H "Authorization: $(echo -n $username:$servicekey |
base64)" \
 -H "API-Key: $apikey" \
 https://app.contrastsecurity.com/Contrast/api/ng/$orgid/
orgtraces/filter?expand=request | \
 jq -r '.traces[] | {uuid: .uuid,
protocol: .request.protocol} | [.uuid, .protocol] | @csv'
```

## Find CWEs associated with CVEs

If you want to find a Common Weakness Enumeration (CWE ) that's associated with a Common Vulnerabilities and Exposures (CVE), use the National Vulnerability Database (NVD) .

While many CVEs have associated CWEs, some might not be classified under a specific CWE or may be associated with multiple CWEs

## Steps

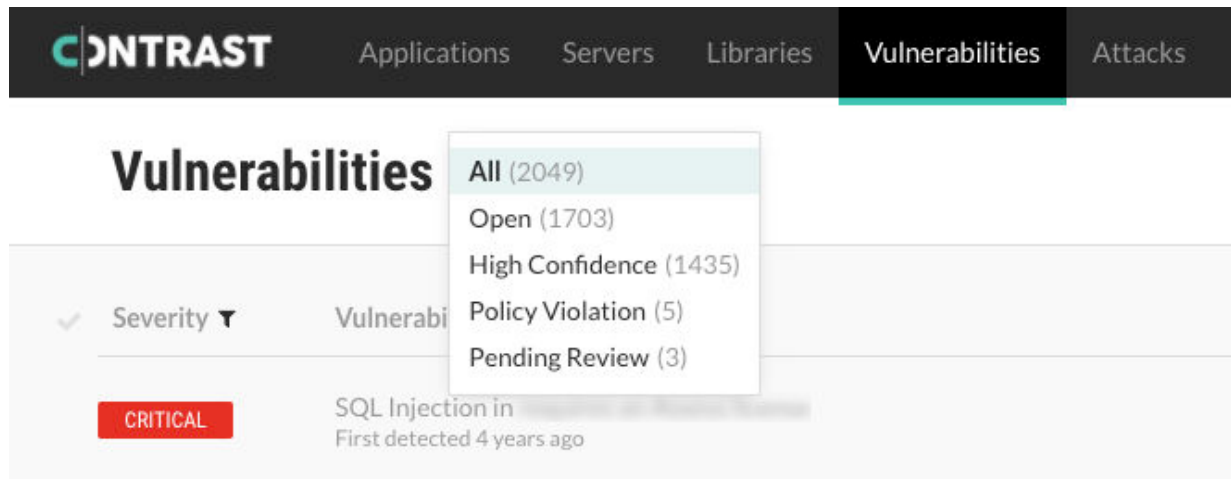
1. Navigate to the [NVD website](#).
2. Search for the CVE.
  - a. From the NVD menu, select **Search**.
  - b. Select **Vulnerabilities - CVE**.
  - c. In Keyword search, enter the CVE identifier (for example, **CVE-2020-12345**).
  - d. Select **Search**.
3. In the search results, select the CVE link.
4. To view the CWE details, under Weakness Enumeration, select any of the CWE links in the displayed list.  
Selecting a CWE link displays the CWE details in the [CWE website](#).

## Vulnerability status

Vulnerability status is shown in the vulnerabilities grid and can be any of the statuses shown in this table. You can edit the vulnerability status.

Status	When to set this status
<b>Reported</b>	This is the default status of a vulnerability after it is discovered by Contrast. The vulnerability in this application could possibly be exploited.
<b>Confirmed</b>	Confirm that the vulnerability is a true finding by reviewing the source code or exploiting it.
<b>Suspicious</b>	The vulnerability appears to be a true finding based on the details provided, but requires more investigation to determine its validity.
<b>Not a problem</b>	<p>The vulnerability is being accounted for without any source code changes. To set this status, you must select one of these reasons. Vulnerabilities set to this status will not revert back to <b>Reported</b> if found again.</p> <ul style="list-style-type: none"> <li>• <b>Attack is defended by an external security control:</b> There is another component in the environment, such as a WAF, which will prevent this vulnerability from being exploited.</li> <li>• <b>False positive:</b> This vulnerability was reported incorrectly. <a href="#">Contact Support</a> to figure out why Contrast flagged this trace as a vulnerability.</li> <li>• <b>Goes through an internal security control:</b> There is custom, corrective code inside the application that will prevent this vulnerability from being exploited.</li> <li>• <b>URL is only accessible by trusted power users:</b> This vulnerability may only exist in specific environments, such as test, and may not exist in production environments.</li> <li>• <b>Other:</b> Select this option if there is another reason that no source code changes are required in order to fix this vulnerability. It is possible to <a href="#">replace Other with a custom value (page 1037)</a> that explains why the vulnerability is <b>Not a problem</b>.</li> </ul>
<b>Remediated</b>	The vulnerability has been fixed by changing source code or config files within the application.
<b>Fixed</b>	The vulnerability has been fixed by changing the source code or because of a reason given under the <b>Not a problem</b> status. A vulnerability set to this status will not revert back to <b>Reported</b> if found again. (This option is only available to administrators.)
<b>Remediated-Auto-verified</b>	This status can only be automatically set. (It can't be manually set by a user.) If a vulnerability is not reported within the time limit set in the <a href="#">vulnerability policy (page 1124)</a> , it will automatically be set to <b>Remediated-auto-verified</b> .

Policies that are set to **Reported**, **Confirmed**, **Suspicious** are considered to be open. Policies that are set to **Not a problem**, **Remediated**, **Fixed**, or **Remediated-Auto-verified** are considered to be closed. You can filter vulnerabilities by **Open** to see only open statuses, or by **All** to see both open and closed statuses.

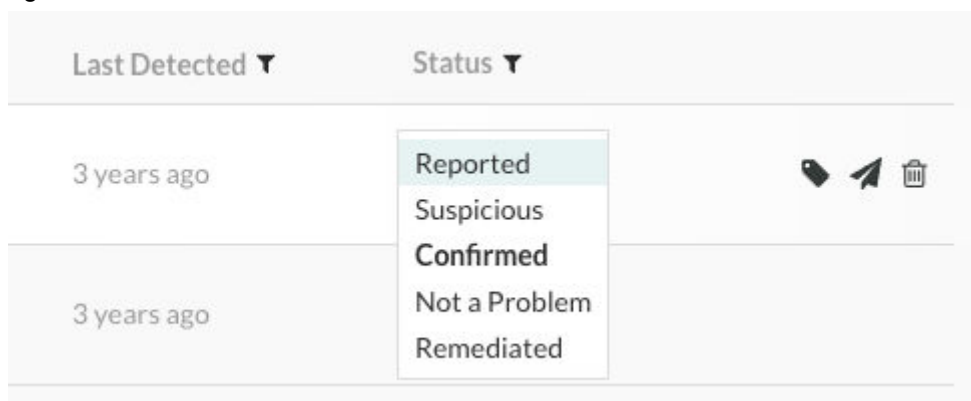


If the agent reports a vulnerability and Contrast has never seen it before, Contrast creates a new entry for the vulnerability. If that vulnerability already exists, Contrast updates the existing entry, issue count and number of days since it was last detected. All vulnerabilities will be reopened with the same pre-existing status, except those that were previously set to **Remediated** or **Remediated-Auto-verified**. Those will be reopened as **Reported**.

### Edit vulnerability status

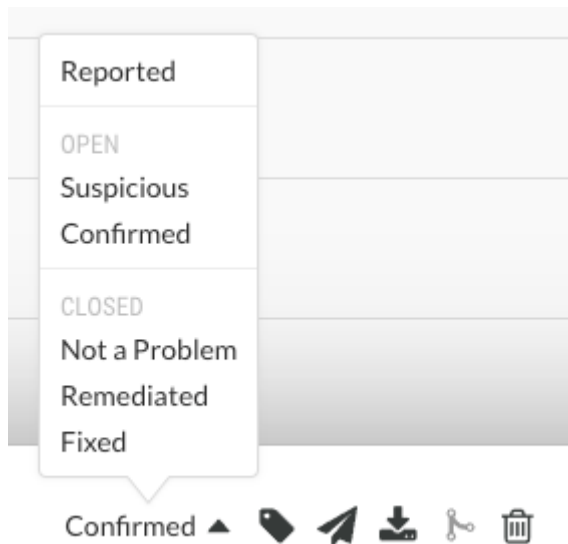
To change the status of one or more vulnerabilities:

1. Select **Vulnerabilities** in the header.
2. To edit a single vulnerability status, find the vulnerability you want to edit and select the status in the **Status** column. You can also change the status from a vulnerability's overview page in the top right corner.



To edit multiple vulnerabilities at a time, use the check marks in the left column to select the vulnerabilities you want to edit. In the batch action menu that appears at the bottom of the page, you will see the current status. Click to expand the menu.





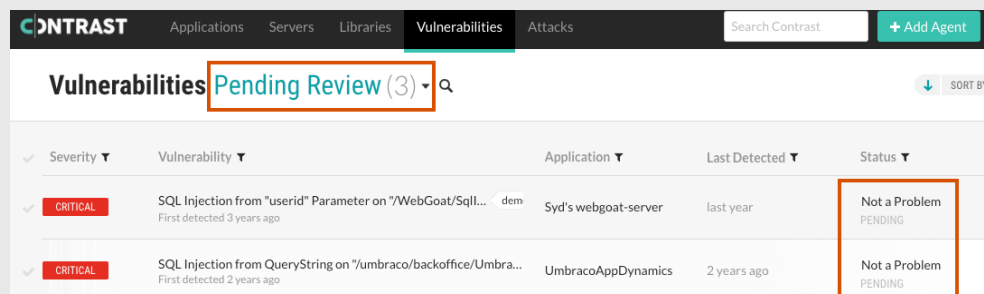
3. Select a new status according to [these criteria](#) (page 1035).



#### NOTE

An Organization Administrator can [require approval](#) (page 1178) before some vulnerabilities can be closed.

If this is the case, you will be required to submit a reason for your status change, and it will be set to **Pending** until it can be reviewed by an Organization Administrator or a RulesAdmin. Hover over **Pending** in the **Status** column to see the date it was submitted.



You may change the status of a pending vulnerability. If approval isn't required for the new status, the vulnerability is no longer marked as **Pending**.

You will receive a notification when your status change request is approved or denied by an administrator. If denied, the vulnerability will go back to its previous state; but, the administrator must provide a reason for the decision. That reason will appear in the vulnerability's **Activity** tab.

4. In the window that appears, select a reason (in the case of **Not a problem**) and enter an explanation for the status change. It is possible to [set a custom reason](#) (page 1037) that vulnerabilities are **Not a problem**.

### Set a custom reason that vulnerabilities are Not a problem

Security teams may determine that a specific vulnerability does not need to be remediated with a code change and set the vulnerability status to **Not a problem**. This helps teams focus on fixing vulnerabilities and prevents Contrast from reporting these vulnerabilities again.

When you use **Not a problem** as a vulnerability status, you must select a reason. Contrast provides [standard reasons \(page 1035\)](#) as well as an **Other** option.

You can change the label **Other** to a value that is meaningful to your organization. To do this:

1. Go to **Policy management** settings for your organization.
2. Select **Vulnerability management**.
3. Select **Set a custom label for Other**.
4. Enter the reason you prefer. This is limited to 25 characters.
5. **Save** your change.

The screenshot shows the 'Policy management' page in the Contrast application. On the left is a sidebar with categories: ASSESS (Assess Rules, Security Controls, Vulnerability Management), PROTECT (Protect Rules, CVE Shields, Virtual Patches, Log Enhancers, IP Management), and GENERAL (Application Exclusions, Compliance Policy). The 'Vulnerability Management' option is selected. The main content area is titled 'VULNERABILITY BEHAVIOR' and contains two sections. The 'Approval workflow' section has a checked checkbox 'Require administrator approval when closing vulnerabilities' and two dropdown menus: 'Remediated' and 'All Severities'. The 'Not a Problem status options' section has a checked checkbox 'Set a custom label for Other' and a text input field containing 'Risk accepted' with a '12 characters left' indicator. A green 'Save' button is at the bottom right.

Now, when marking vulnerabilities as **Not a problem**, the values listed will include the custom reason instead of **Other**.



## NOTE

When you change **Other** to a custom label or change it back to **Other**, all the vulnerabilities with that label will change to the new label for your organization.

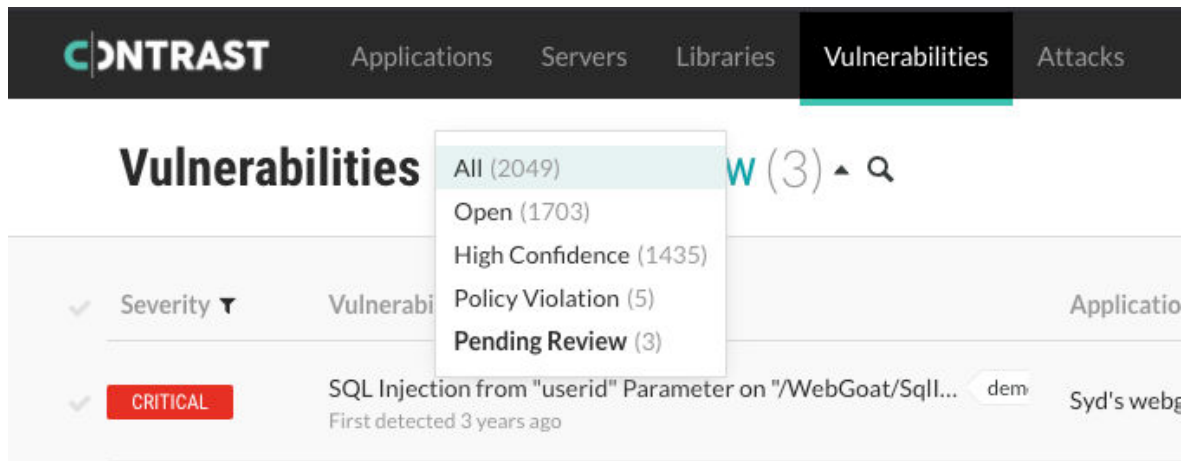
## Review pending vulnerability status changes

If an Organization Administrator has [required approval for a particular vulnerability \(page 1178\)](#), the [status \(page 1035\)](#) won't change until it is approved. This can apply to manual vulnerability status changes, two-way bugtracker integrations, as well as auto-verification policies.

You must be an Organization RulesAdmin with RulesAdmin permissions for the target application in order to approve or deny vulnerability closures.

To do this:

1. Select the link in your notification in the Contrast application, or select **Vulnerabilities** in the header, then select the filter at the top of the grid to view all pending reviews.



2. Use the check marks in the left column to select one or more vulnerabilities. In the batch action menu that appears at the bottom of the page, select **Review**. Then select **Approve** or **Deny**. You can also select **Review** in the top right from a vulnerability overview page.
3. If you deny the status change, you must provide a reason. Denied vulnerabilities revert to their previous status. Approved vulnerabilities take the new status and are no longer marked **Pending**. Either way the results of the review will display in the vulnerability's **Activity** tab.

### Edit vulnerability severity

Contrast classifies vulnerabilities in an application into five severity levels. The classifications are based on the likelihood and impact of a vulnerability in the application, from most to least severe:

- Critical
- High
- Medium
- Low
- Notes

To change a vulnerability's severity level:

1. Select **Vulnerabilities** in the header.
2. Select the colored badge in the **Severity** column and choose a new level from the menu. (You cannot update the severity of multiple vulnerabilities at once.)

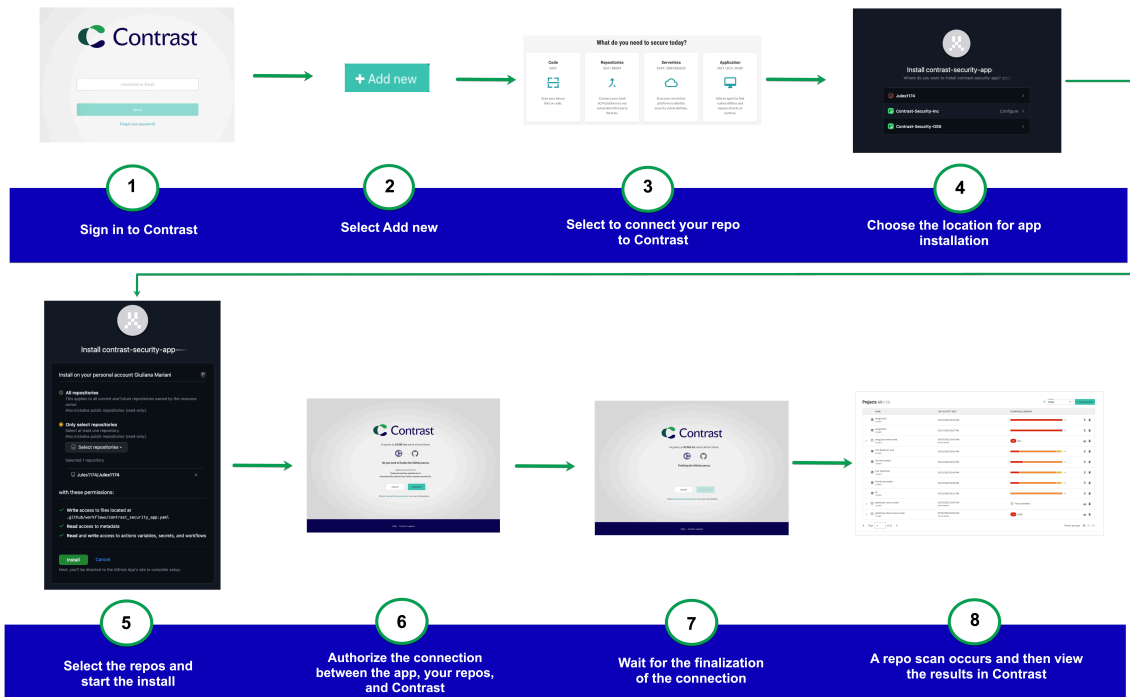
## Contrast Security GitHub App

Use the Contrast Security GitHub App (also known as **Contrast Security SCA** in the GitHub Marketplace) to scan GitHub repositories with Contrast. Detect vulnerable libraries with how-to-fix guidance, and automate your CI/CD to prevent risk, at an earlier step, in your team's code.

### How it works

For first-time use, sign in to Contrast, connect your GitHub account to Contrast and scan for library vulnerabilities in a repository.

Click the GitHub icon  to use the [Contrast Security GitHub App \(page 936\)](#) to connect with Contrast.



Once connected and scanned you can view the results in the [Projects \(page 600\)](#) list in Contrast.

You can also [connect from the GitHub Marketplace \(page 1042\)](#) with the Contrast Security GitHub App.

With this app, you can:

- Scan a GitHub repository
- Automate the security analysis of dependencies so that vulnerabilities can be detected and resolved during code review rather than after detection or exploitation in testing or production environments
- Any commits to the default branch and PRs created to merge into the default branch will trigger the workflow file. In addition, you can manually trigger the workflow.
- Users with edit, rules admin, or admin permissions will have access to the app

## Next steps

- [Install and authorize \(page 1041\)](#)
- [GitHub repository connections \(page 1042\)](#)
- [Troubleshoot \(page 1043\)](#)

## Contrast SCA supported languages

Contrast's SCA function supports the following languages:

### Runtime

Language	Sources
Java	maven central, redhat GA
.NET	Nuget, framework/core downloads
Node.js	npm
PHP	composer (packagist), Wordpress packagist
Python	pypi
Ruby	rubygems
Go	go index

## CLI and SCA in repository

For further details on the supported versions for CLI see the [Contrast CLI supported languages and package managers \(page 995\)](#) page.

Language	CLI	SCA in repository
Java - Maven	✓	✓
Java - Gradle	✓	✗
.NET	✓	✗
Node	✓	✓
PHP	✓	✓
Python	✓	✓
Ruby	✓	✓
Go	✓	✓

## Installation and authorization

Connect your GitHub account to see vulnerable third-party libraries. Once connected, you can monitor PR summaries and triggers for analysis and view them here in Contrast.



### NOTE

There is another method for monitoring vulnerabilities in Contrast. [Integrate Contrast with Github \(page 1080\)](#) to view vulnerabilities in projects.

## Contrast and GitHub secrets

Contrast cannot read GitHub secrets even though the screen contains READ/WRITE secrets. This is only a readout of the token name.

You will need the following credentials from your Contrast account to create the secrets in GitHub. You can find them in Contrast under **user menu > User settings > Profile**.

- API key (CONTRAST\_API\_KEY)
- Organization ID (CONTRAST\_ORGANIZATION\_ID)
- Authorization header (CONTRAST\_AUTH\_HEADER)

You will also need the address of the Contrast installation to which your agent would like to report (CONTRAST\_API\_URL). Defaults to: <https://app.contrastsecurity.com>.

The Contrast Security GitHub App creates repository secrets and action variables for use in the workflow so results are sent to the correct Contrast account. Closing a PR requires manually deleting these secrets and variables. You can find the secrets and variables under the `/settings/secrets/actions` page of your GitHub account.

## Before you begin

- Make sure you have access to the Contrast web interface.
- Make sure you are logged in to your GitHub account.

## Steps

### To install from within Contrast:

1. Log in to Contrast and select **Add New** at the top right.
2. Select the **Connect GitHub** option and click **Next**.
3. On the install screen, select either to connect all the repositories or specific repositories.
4. Enter the URL of your Contrast host domain. For example, <https://app.contrastsecurity.com>.
5. Select **Install** and wait for the connection to be established between Contrast and GitHub.
6. Select **Authorize** to finalize the connection and to onboard your repositories.

Once complete, you can view your repository analysis results on the [Projects \(page 600\)](#) list.

### To install from the GitHub Marketplace:

1. Go to the [Contrast Security GitHub app](#) on the GitHub Marketplace.
2. Select **Install it for free**.
3. Follow the steps to install and connect Contrast with your repositories.

Once complete, you can view your repository analysis results on the [Projects \(page 600\)](#) list.

## Add or disconnect GitHub repositories

Add repositories to Contrast or disconnect repositories from Contrast.



### NOTE

An Organization Editor role is required at minimum to be able to add repositories.

## Add a repository

You can add additional repositories in Contrast with the **Add repository** button under the Projects view in Contrast or through GitHub.

### To add a repository through Contrast:

1. Select **Projects** in the header.
2. Click the **Add repository** button in the upper-right of the window.
3. Log in to your GitHub account and select the repository you would like to connect to Contrast.

Once added, Contrast performs a scan of the repository and provides results in the [Projects \(page 600\)](#) list.

### To add a repository through GitHub:

1. In Github, go to **Settings > Applications > Integrations** and locate the **Contrast Security GitHub App**.
2. Under **Repository access** select the repositories you want to connect to Contrast.

3. Click **Save**.

Once added, Contrast performs a scan of the repository and provides results in the [Projects \(page 600\)](#) list.

## Disconnect repositories

To disconnect repositories from the Projects list in Contrast:

1. Select **Projects** in the header and locate the row in the grid for the project you want to disconnect.
2. Click the **Delete** icon under the actions column.

This action disconnects the repository and all associated projects in it. Note that this **does not** remove the repository from the Contrast Security GitHub App installation. If you also wish to remove Contrast's permissions to this repository, you will need to remove access through the GitHub App installation settings **after** disconnecting the repository here to ensure proper cleanup.

Remove access to your repositories and uninstall the Contrast Security GitHub App through GitHub. In Github, go to **Settings > Applications > Integrations** and locate the **Contrast Security App**. Select **Uninstall**.

## Troubleshoot

If you cannot complete installation and onboarding through the GitHub App, try these options.

- **Look at cookies.** Clear the `ONBOARDING_SESSION` cookie manually
- **Reinstall the app.** Close the browser and uninstall the app; then reinstall the app

If you are unable to run any workflows:

- **Check Actions permissions in GitHub.** Ensure the **Disable actions** option is not selected under the permissions tab of your GitHub account settings. If the option is selected, the workflow will not run and perform the analysis.

## Reports

All Contrast reports, with the exception of the Software bill of materials, are delivered as time-stamped PDFs that are downloaded locally.

These reports are available:

- [Attestation reports \(page 1043\)](#)
- [DISA STIG viewer checklists \(page 1045\)](#)
- [Software bill of materials \(page 1046\)](#)
- [Remediation summary report \(page 1049\)](#)
- [Vulnerability trend report \(page 1047\)](#)
- [Organization statistics \(page 1049\)](#)
- [Report dashboard \(page 1318\)](#) (aggregated vulnerabilities)

## Attestation reports

Attestation reports provide evidence of vulnerability remediation based on the most current application information. Meet compliance and auditing requirements with these PDF reports.

As of November 7, 2023, this report replaces the Security standards report. The Attestation report provides similar information as the Security standards report. It will help you meet compliance requirements and Identify areas of urgent attention.

**NOTE**

This report expires seven days after you create it. Contrast deletes the report after this time.

Attestation reports include:

- An itemized list of the specific filter settings used to run the report
- A summary of the security posture for the application
- Vulnerabilities assessment for both custom code and open-source libraries. Note that critical severities will not be displayed in this section if CVSS 3.1 has not been turned on for existing organizations. To enable this, contact [Contrast Support](#).
- Route coverage as a security assessment metric
- An optional compliance policy assessment and detailed information about open vulnerabilities for the application
- An appendix that describes methodologies and terminologies

**Before you begin**

An Attestation report has the following limits:

- 1,350 vulnerabilities with details
- 18,000 vulnerabilities without details
- 15,000 routes with observations
- 30,000 routes without observations

If your report exceeds these limits, an error message displays and the report doesn't generate. If this situation occurs, change your report selections to reduce the amount of information in the report.

**Steps to generate an Attestation report**

1. Select **Applications** in the header.
2. Select an application in the Applications grid.
3. Select the **Reports** icon (■) located at the top-right of the application's page.
4. Select **Generate Attestation Report** from the list.
5. In the Attestation report window, select the **Vulnerabilities**, **Environments**, and additional **Security Standards** that you want to include in the report.



**Attestation Report** ×

Attestation reports provide evidence of vulnerability remediation based on the most current application information. Meet compliance and auditing requirements with these PDF reports.

**Vulnerabilities**

All Rules (83)

☐ Include vulnerability details  
☐ Include route observations

**Environments**

All Environments (3)

**Security Standards**

Choose standards...

Note: Applying security standards will result in the inclusion of an additional security standards section in the generated report.

📘 Total routes and vulnerabilities: 0 routes and 1 vulnerabilities

Cancel Generate

The default is to show all vulnerabilities and environments, but you can filter them by selecting the fields and then, selecting filters. Choose an option from **Security Standards** to include an additional Security Standards section in the generated report.

Optionally, you can choose to include detailed information about open vulnerabilities and observed routes.

The following table includes the categories that you can use to create a custom report.

Category	Default	Filter options
Vulnerabilities	All	<ul style="list-style-type: none"><li>• Status (Reported, Suspicious, Confirmed, Not a Problem, Remediated, Fixed, Remediated - Auto-Verified)</li><li>• Severity (Note, Low, Medium, High Critical)</li><li>• Assess Rules</li></ul>
Vulnerability details	None	Include vulnerability details by selecting the checkbox for it.
Route observations	None	Include details about observed routes by selecting the checkbox for it.
Environments	All	<ul style="list-style-type: none"><li>• Development</li><li>• QA</li><li>• Production</li></ul>
Security Standards	None	<ul style="list-style-type: none"><li>• DISA ASD STIG</li><li>• IPA-7.0</li><li>• OWASP 2013 Top 10</li><li>• OWASP 2017 Top 10</li><li>• OWASP 2021 Top 10</li><li>• OWASP Top 10 API Vulnerabilities 2019</li><li>• PCI DSS - 2.0</li><li>• PCI DSS - 3.0</li><li>• PCI DSS - 3.2.1</li><li>• PCI DSS - 4.0</li></ul>

6. Select **Generate**.

After Contrast generates the report, the **Notifications** panel displays a download link for it. Select the link to download the report.

## DISA STIG Viewer checklists

DISA's Security Technical Implementation Guide (STIG) is the basis for evaluation of the security of all government applications. The STIG is intended to be used throughout the life cycles of these applications in order to provide security assurance for these applications. Contrast's compliance

reporting can provide a listing of the vulnerabilities found in your application that violate guidelines of multiple STIGs.



### IMPORTANT

An application must have an Assess license to run a DISA STIG report.

Before DISA STIG reports can be run, a SuperAdmin must enable it. Select **SuperAdmin** in the user menu, then select **Organizations** in the header. In the window that appears, select the box to **Enable DISA STIG Checklist reporting** and select **Save**.

STIG Viewer creates custom checklists with multiple STIGs for compliance reporting. You must import your application's checklist to get the DISA STIG report on those vulnerabilities from Contrast.

To run a STIG Viewer checklist:

1. Go to the **Applications** page and select an application.
2. In the application's **Overview** page, click the reporting icon and select **Generate STIG Viewer Checklist**.
3. In the window that appears, import a STIG Viewer checklist (.ckl) file. This file must be a checklist exported from the STIG Viewer application.
4. Click **Generate** to download an updated STIG Viewer checklist (.ckl) file.

## Software bill of materials (SBOM)

A Software Bill of Materials (SBOM) might be required for compliance with government security regulations.

You can generate an SBOM through Contrast, through a simple API, or with a command through the Contrast command line interface (CLI).

The Contrast SBOM meets the specifications of the OWASP's CycloneDX SBOM standard and the international open SPDX standard. It contains information about the software that your application uses including:

- Libraries - Open source and third-party components present in a codebase
- Licenses that govern the software components
- Versions of software components used in the codebase



### NOTE

Currently supports CycloneDX 1.6 and SPDX 3.0.


The Contrast SBOM also meets the requirements of the National Telecommunications and Information Administration (NTIA). It includes the author name, supplier name, component name and version, component relationship, timestamp and other unique identifiers like PURL and package SPDX identifier.

## Before you begin

- A Contrast Assess license is required for export via Contrast
- Supported languages: Java, .NET Framework, .NET Core, Node.js, Python, Ruby, Go, PHP

## Steps

There are three options for generating an SBOM report.

1. **To generate a report with Contrast:**
  - a. Select **Applications** in the header.
  - b. Select an application name from the list to view the application's **Overview** tab.
  - c. Select the **Reports** icon (  ) located at the top-right of the application list.
  - d. In the dropdown, select **Generate Software Bill Of Materials (SBOM)**
  - e. Select the CycloneDx or SPDX option in the dialog and select the **Generate** button to generate and download a copy of the SBOM.
2. **To generate a report with API:**
  - a. For CycloneDX: Make a **GET**<HOST>/Contrast/api/ng/<ORG\_ID>/applications/<APP\_ID>/libraries/sbom/cyclonedx request.
  - b. For SPDX: Make a **GET**<HOST>/Contrast/api/ng/<ORG\_ID>/applications/<APP\_ID>/libraries/sbom/spdx request.See [REST API](#) for more information about using APIs.
3. **To generate a report with CLI:**
  - Use the `--save` command. Choose the type with `--save cyclonedx` or `--save spdx`. See [CLI commands \(page 1003\)](#) for more information.



### NOTE

- .NET support is currently limited for CLI.
- Use the CLI to generate the SBOM for [static SCA \(page 924\)](#) results.
- The SBOM generated via CLI will provide class usage information of the application the CLI is registered to that has library data.

## Vulnerability trend reports

Use the vulnerability trend reports to recognize the vulnerabilities your applications face and how well they're being managed.

To view vulnerability trend reports:

1. Select **Reports** in the user menu. Select **View** to see the graphs in more detail.
2. Select **New** to see a graph of new vulnerabilities. Select **Total** to see a graph of all reported vulnerabilities compared to all remediated vulnerabilities.  
Each black data point represents the total number of Suspicious, Confirmed and Reported vulnerabilities for that date. Each green data point represents the total number of vulnerabilities marked as **Not a problem**, **Remediated** or **Fixed**. Hovering over each data point generates a tooltip with status breakdowns.
3. Each report defaults to all applications, servers and rules. Filter vulnerabilities by clicking in the fields above the graph. The following table outlines the categories that you can use to create a custom report.

Field	Default	Filter options
Date	Last 7 days	Last 30 days Last 12 weeks Last 12 months
Applications	All	Importance (Critical, High, Medium, Low, Unimportant) Application Tags Licensed (List of all applications)
Servers	All	Environment (development, test, production) Server Tags Servers (List of all servers)
Rules	All	Severity (Critical, High, Medium, Low, Note) Vulnerability Tags Vulnerability Rules (List of all rules)

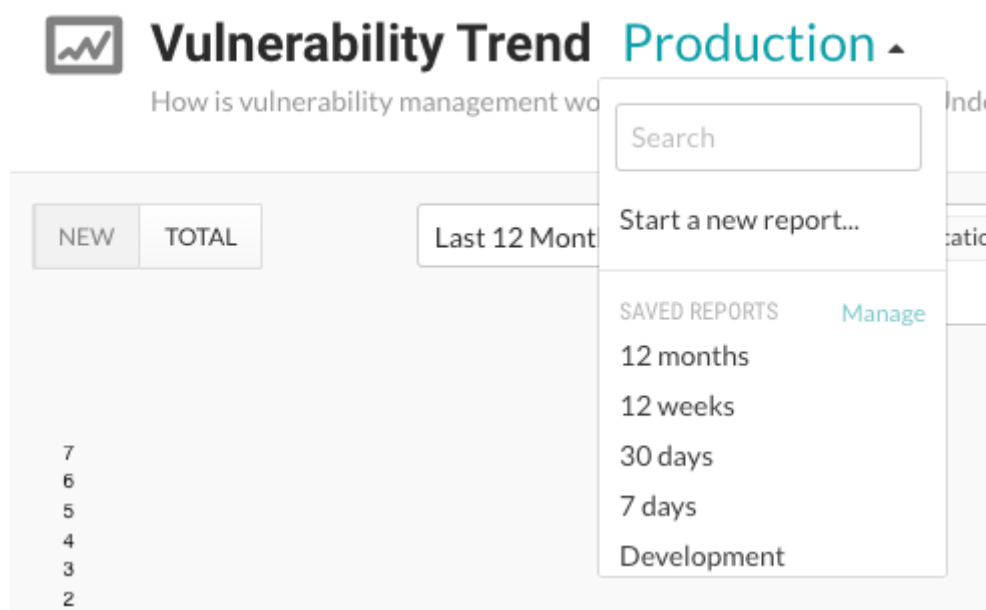
4. Select **View**, then select the **Save report** icon in the top right to save this filter criteria to a report. Name the report in the window that appears and select **Save**. Saved reports are unique to the user, so you have your own defined list of saved vulnerability trend reports. You can edit or delete these reports at any time.

If you change filter options while viewing a saved report, the star icon changes to an unsaved state and **Edited** appears next to the report name. You can then use the same icon to **Save existing** or **Save as new**. Choose **Save existing** to update the saved report name with the current filters and remove the **Edited** status. Choose **Save as new** to save the report view with the current filters as a new report under a different name.

Click **Remove** to permanently delete the saved report that you're currently viewing.

To clear unsaved edits to an existing report and start over with the report defaults, choose the **Start a new report** option in the dropdown.

When you've created more than five saved reports, you will see a **Manage** link in the **Saved reports** dropdown. Select **Manage** to open this window. Here you can rename reports (click on the name to edit), search for reports, or use the check boxes next to each report (or use the **Select all** check box) to select which ones you want to remove.



5. You can also create a timestamped PDF export of the Vulnerability Trend to capture a snapshot of your vulnerability management. Select the **Export** icon in the upper right hand corner of the page. Contrast downloads the report to your desktop. Each PDF report includes a summary of

the variables included in your customized view, the trend graphic, and a table of the metrics and breakdowns of each data point.

## Organization statistics

Select **Reports** in the user menu to find [vulnerability trends \(page 1047\)](#), as well as organization level information on:

- **Licenses:** View the number of overall licenses for Assess and Protect, as well as the number of unlicensed applications and servers that exist in your organization.
- **Applications:** The inner ring designates the breakdown by language. Choose the categories you want to compare in the outer ring by selecting **Technology** or **Grade** in the dropdown.
- **Servers:** Select **Container** or **Environment** in the dropdown to choose how the numbers are analyzed.

Use the filters in the dropdowns to choose which data to compare at a glance.

Select **View** for more details including:

- **Licenses:** Under **Activity**, view an activity trend chart of data on license consumption over the past year.  
Hover over a data point on the Assess or Protect trend lines to see how many licenses were used each month. The dotted line shows the number of licenses purchased.  
Click on the vertical bars in the chart to view your hourly usage of Protect licenses for each day. Peak hourly usage is represented by bright green shading at the top of the bars. Select **Back to license activity** to return to your view of license activity data.  
Select **View Protect usage** below the activity chart to see data for the current month and a usage statistics. Use the dropdown to view data for a different month.  
Under **Consumption**, you can see a thermometer chart and a timeline for Assess and Protect. The thermometer chart shows the total number of licenses purchased compared to the number being used. The timeline shows how many licenses are about to expire on given dates.  
The circular charts on the right show breakdowns by fraction and percentage for Assess and Protect. If your organization doesn't own any Protect or Assess licenses, the chart shows the count of unlicensed assets.
- **Applications:** Under **Status** you can see the total number of applications broken down by the number that are licensed, unlicensed and archived, as well as how many licenses are available in your organization.  
The circular Language Breakdown chart shows the number of applications, broken down by language in the inner band, and by technology in the outer band. Hover for more details.  
Under **High risk** and **Expirations** you can see the number of applications with critical open vulnerabilities and expiring licenses.  
Under **Protection coverage** you see the number of applications on production servers that have incomplete coverage. Select **View breakdown** for more details.  
Applications that were added within the last week and applications that reside on an offline server are listed separately in the sidebar.
- **Servers:** Under **Environments**, you can see all deployed servers by environment.  
Under **Container breakdown** you can see the number of deployed servers for each language in a given environment. Use the dropdown to view data for a different environment.  
Under **Snapshots**, you can see the number of servers with Assess and Protect enabled, as well as all servers online compared to the total number of servers in the given environment.  
The right sidebar includes a list of new, offline, deleted and expiring servers.

## Remediation summary package ☹

The remediation summary package contains a set of charts, in PDF format, that help you determine how well your remediation process is working. The Remediation summary section on the Reports page has a download button that lets you download the report package.

Contrast updates the charts on a monthly basis. This date is displayed below the Download charts button.

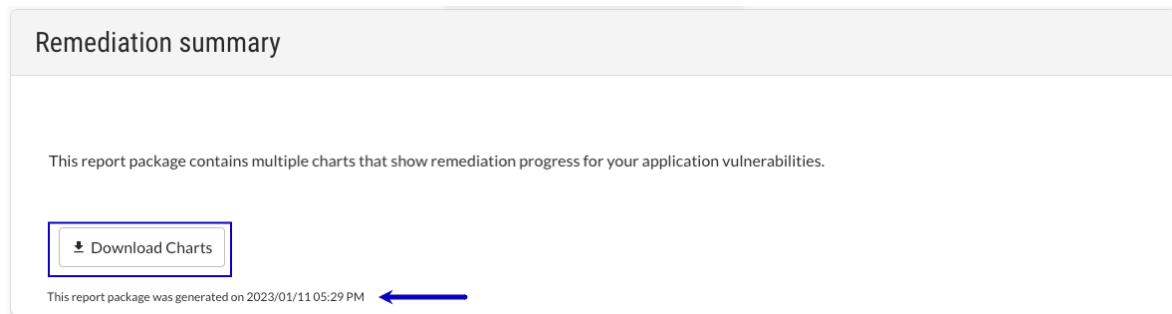
The report package also includes a `README` file that provides guidance for interpreting the charts.

## Before you begin

- [Diagnostic reporting \(page 1250\)](#) must be enabled.  
The Reports page does not display the Remediation summary section if diagnostics are disabled.
- The View applications action is required.

## Steps

1. From the user menu, select **Reports**.
2. In the Remediation summary section, select **Download charts**.



3. Save the ZIP file in a convenient location.
4. Extract the files and view the `README` file and the individual charts.

# Integrations

Documentation is provided for supported integrations that are part of the core, recommended way that Contrast works. Contrast may be compatible with other tools or scenarios developed by the community that are not supported. For specific information on third-party tools and technologies, consult the documentation for that product. Also, there are additional articles in the [Contrast Support Portal](#) around specific use cases or workarounds.






## NOTE


Note that supported integrations are further documented in the left-hand navigation menu. Community integrations are linked to external documentation. Links to documentation in the Contrast Support Portal or third-party sites are indicated by the [external link icon](#).

You must have the organization administrator role in Contrast to view the integrations available to your organization.

## ADR integrations


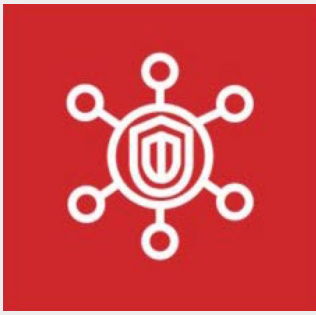



Integrate Contrast ADR with SIEM tools.

	Contrast Security ADR for Splunk	<a href="#">Contrast ADR with Splunk (page 1110)</a>
 DATADOG	Datadog	<a href="#">Source Code</a>
	Sumo Logic	<a href="#">Source Code</a>

	<b>Azure Sentinel</b>	<a href="#">☑ Contrast Protect Azure Sentinel Solution</a> <a href="#">☑ Azure Sentinel</a>
-----------------------------------------------------------------------------------	-----------------------	------------------------------------------------------------------------------------------------

## Platform integrations

Integrate Contrast with your platform tools.


	<b>Amazon Security Lake</b>	<a href="#">Amazon Security Lake with Contrast Assess (page 1064)</a>
	<b>AWS Security Hub</b>	<a href="#">AWS Security Hub with Contrast Assess (page 1063)</a>
	<b>Jira Cloud</b>	<a href="#">Jira Cloud integration with Contrast Scan</a>
	<b>ServiceNow</b>	<a href="#">ServiceNow Integration (page 1106)</a>
	<b>Wiz</b>	<a href="#">Wiz integration (page 1116)</a>

## Integrations



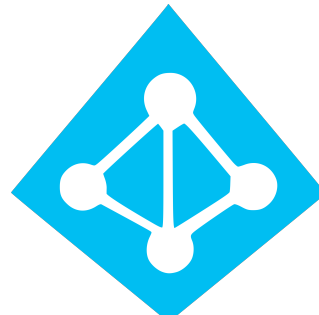

Integrate Contrast with your existing tools to streamline your workflows.

### AWS









	<b>AWS Elastic Beanstalk</b>	<a href="#">Java agent with AWS Elastic Beanstalk (page 137)</a>
-----------------------------------------------------------------------------------	------------------------------	------------------------------------------------------------------

## Azure

	<b>Microsoft Azure</b>	<ul style="list-style-type: none"> <li><input checked="" type="checkbox"/> <a href="#">.NET and Azure App Service</a></li> <li><input checked="" type="checkbox"/> <a href="#">.NET and Azure Arm</a></li> <li><input checked="" type="checkbox"/> <a href="#">.NET and Terraform</a></li> </ul>
	<b>Azure DevOps</b>	<ul style="list-style-type: none"> <li><a href="#">Azure Pipelines Extension (page 1069)</a></li> <li><input checked="" type="checkbox"/> <a href="#">Azure Pipelines Marketplace</a></li> </ul>
	<b>Azure Active Directory</b>	<ul style="list-style-type: none"> <li><input checked="" type="checkbox"/> <a href="#">Azure Active Directory single sign-on (SSO) integration</a></li> </ul>
	<b>Azure Boards</b>	<a href="#">Azure Boards Integrations (page 1065)</a>








## Contrast SDKs and Webhooks

	<b>Contrast CLI</b>	<a href="#">Contrast CLI (page 994)</a> <a href="#">NPM</a>
	<b>Java SDK</b>	<a href="#">Contrast Java SDK</a>
	<b>JavaScript SDK</b>	<a href="#">Contrast SDK Javascript</a> <a href="#">NPM</a>
	<b>.NET SDK</b>	<a href="#">Contrast SDK .NET</a> <a href="#">nuget</a>
	<b>Python SDK</b>	<a href="#">Contrast SDK Python</a>

	<b>Webhooks</b>	<a href="#">Generic Webhooks (page 1076)</a>
-----------------------------------------------------------------------------------	-----------------	----------------------------------------------

## GitHub and GitHub actions



Learn about the different ways you can [integrate Contrast technologies with GitHub \(page 1081\)](#)

	<b>Github</b>	<a href="#">GitHub Integration (page 1080)</a>
	<b>Contrast Amazon Elastic Kubernetes Service Build Deploy</b>	<a href="#">☑ Contrast Amazon EKS Build Deploy</a>
	<b>Contrast Azure Kubernetes Service Build Deploy</b>	<a href="#">☑ Contrast AKS Build Deploy</a>
	<b>Contrast Azure Spring Cloud Deploy</b>	<a href="#">☑ Azure Spring Cloud Deploy</a>
	<b>Contrast SCA Action</b>	<a href="#">☑ Contrast SCA Action</a>
	<b>Contrast Scan Analyze</b>	<a href="#">☑ Contrast Scan Analyze</a>
	<b>Contrast Verify</b>	<a href="#">☑ Contrast Verify</a>



## Jira

	Jira	<a href="#">Jira Integration (page 1097)</a>
-----------------------------------------------------------------------------------	------	----------------------------------------------

## Microsoft Teams and Slack

	Microsoft Teams	<a href="#">Teams Integration (page 1104)</a>
	Slack	<a href="#">Slack Integration (page 1107)</a>

## Cloud integrations

	Google App Engine	<a href="#">☑Configure the Java agent for Google App Engine</a> <a href="#">Google App Engine</a>
	Red Hat OpenShift	<a href="#">☑Contrast Security containers</a>





**VMware Tanzu™****VMware Tanzu**  
(formerly Pivotal Cloud Foundry)[Configure Java for VMware Tanzu \(page 132\)](#)[Configure Node.js for VMware Tanzu \(page 345\)](#)


## Continuous integration and build tools

**Bamboo**[Bamboo Plugin \(page 1073\)](#)[☑ Contrast Security for Bamboo Marketplace](#)[☑ Contrast Bamboo Plugin Source Code](#)**CircleCI**[☑ Orb Registry](#)[☑ Contrast Security Orb for CircleCI](#)**GitLab****GitLab**[☑ How to integrate Contrast in a GitLab pipeline](#)**Gradle**[Gradle Plugin \(page 1084\)](#)[☑ Gradle.org and Contrast Plugin](#)[☑ Contrast Gradle Plugin Source Code](#)[☑ Sample Project](#)**Jenkins**[Jenkins Plugin \(page 1089\)](#)[☑ Contrast Continuous Application Security](#)[☑ Jenkins Source Code](#)




	<b>Maven</b>	<a href="#">Maven Plugin (page 1103)</a> <a href="#">☑Maven Central Repository</a> <a href="#">☑Contrast Maven Plugin Source Code</a> <a href="#">☑Sample Project</a>
-----------------------------------------------------------------------------------	--------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## IDE plugins



	<b>Eclipse</b>	<a href="#">☑Contrast Security for Eclipse Marketplace</a>
	<b>IntelliJ</b>	<a href="#">Contrast IntelliJ Plugin (page 1085)</a> <a href="#">☑Contrast Security for IntelliJ Marketplace</a>
	<b>Visual Studio</b>	<a href="#">Visual Studio Plugin (page 1114)</a> <a href="#">☑Contrast for Visual Studio Marketplace</a>
	<b>Visual Studio Code</b>	<a href="#">Visual Studio Code Plugin (page 1114)</a> <a href="#">☑Contrast Security Plugin Marketplace</a>


	<p><b>Visual Studio for Mac</b></p>	<p><a href="#">Visual Studio for Mac Plugin (page 1116)</a></p> <p><a href="#">Extension file</a></p>
-----------------------------------------------------------------------------------	-------------------------------------	-------------------------------------------------------------------------------------------------------

## Incident management systems

	<p><b>PagerDuty</b></p>	<p><a href="#">PagerDuty Integration (page 1105)</a></p>
	<p><b>Splunk on-call</b> (formerly VictorOps)</p>	<p><a href="#">VictorOps Integration (page 1113)</a></p>
	<p><b>Splunk</b></p>	<p><a href="#">How to integrate Contrast with Splunk</a></p>

## Work tracking platforms

	<p><b>Kenna Security</b></p>	<p><a href="#">Kenna toolkit</a></p> <p><a href="#">Running Contrast Security tasks</a></p>
	<p><b>Rally</b> (formerly CA Agile Central)</p>	<p><a href="#">Agile Central Integration (page 1061)</a></p>

	<b>Solutions Business Manager</b> (formerly Serena)	<a href="#">Serena Business Manager Integration (page 1105)</a>
	<b>ThreadFix</b>	<a href="#">☑ Contrast Remote Provider (ThreadFix)</a> <a href="#">☑ Contrast Remote Provider</a>

## Integration release notes

### April 2025

**Release date:** April 15, 2025

#### New and improved:

- Revised and improved the [Contrast Azure Pipelines \(page 1069\)](#) extension. (PROD-3504)
- Updated the [Contrast Azure Boards \(page 1065\)](#) integration to include options for [excluding sensitive data in tickets \(page 1068\)](#), [bi-directional integration, \(page 1067\)](#) [managing credentials \(page 1066\)](#) in Contrast configurations, and sending Contrast session metadata to tickets. (PROD-3144, PROD-3143, PROD-3114, and PROD-3370)
- Updated the Java agent to support Maven 3.9.9 and the Gradle plugin. (PROD-2855)
- Added a link to the Contrast AI guidance in the How to Fix information sent to Jira tickets. (PROD-3431)
- Added a link to the Contrast AI guidance in the How to Fix information sent to Azure Boards tickets. (PROD-3576)

#### Bug fixes:

- Fixed an issue that caused an interruption with a Jira connection following a maintenance window. (PROD-3513)

### March 2025

**Release date:** March 18, 2025

#### New and improved:

- Contrast ADR has a new API-based [integration with Splunk \(page 1110\)](#) to provide timely actionable attack exploit events across your entire application portfolio. (PROD-3269)

### February 2025

**Release date:** February 20, 2025

#### New and improved:

- **Preview:** Update [Visual Studio Code IDE plugin \(page 1320\)](#) to support SAST. (PROD-3112)
- Update to [IntelliJ IDE support \(page 1085\)](#) for static and runtime. (PROD-2753)



## January 2025

**Release date:** January 21, 2025

### New and improved:

- **NEW:** Addition of a [Wiz integration \(page 1116\)](#) to send runtime security information regarding applications from Contrast to a Wiz deployment. (PROD-2694)
- To ensure our customers can continue using MS Teams after the January 2025 Webhook URL deprecation, as Microsoft has already shared [here](#), we have added support for the new power automate integration. This will be enabled by request, and we recommend customers follow the instructions in the [integration documents \(page 1104\)](#) to prepare for the change. Once complete, please [contact Support](#) to enable this integration. Note, if you use the MS Teams integration and do not make these changes by January 31st, the integration will fail. (PROD-3092)
- Added the ability to [synchronize comments \(page 1098\)](#) under the **Activity** tab to link the comments between the **Activity** tab and the Jira issue. Note that the Jira administrator must [register and configure a webhook in Jira](#) using the *Comment...* actions. (PROD-3118)
- Added the ability to [delete expired credentials \(page 1101\)](#). (PROD-3119)
- Users can select the ability to [mask sensitive information being sent to Jira \(page 1098\)](#). Remove sensitive information from the *Issue* title and additional fields when creating the ticket. (PROD-3120)
- Support for the Japanese language was added to the MS Teams Integration. (PROD-3261)
- Added the feature to [show session metadata \(page 1098\)](#) associated with the [vulnerability \(page 1022\)](#) in the Jira ticket. (PROD-3369)

### Bug fixes:

- Fixed a bug that would cause bi-directional status updates to stop working between the Contrast vulnerability record and the Jira ticket. (PROD-3117)

## December 2024

**Release date:** December 10, 2024

### Advance notice regarding Contrast MS Teams Integration

- On January 31st, 2025, the Microsoft Webhook-based connectors within the O365 Connectors service in Teams are transitioning to a new URL structure due to the implementation of further service hardening updates. This will impact the Contrast MS Teams Integration. As a result, we will update the MS Teams integration to support this change as part of our January release. We ask customers using the MS Teams integration to follow this [guide](#) to prepare themselves for the change. More information on our upcoming change can be found on the [Microsoft Teams preview \(page 1104\)](#) page.

## Integrate with Agile Central

Integrate Agile Central with Contrast to automatically track vulnerabilities in your applications.

Before you start, be sure you have:

- An Agile Central account URL.
- Permission to create issues in the target project.
- A running Agile Central instance accessible via HTTP to Contrast.
- A project to associate the application instrumented by Contrast.

To connect with Agile Central:

1. Go to **Organization settings > Integrations** in the **user menu**.
2. Select **Connect** in the Agile Central row.

3. In the **Connect with Agile Central** form, add the name for the bugtracker entry, as well as the **URL** and **API Key** in the given fields. The Agile Central URL must be accessible from Contrast instance being configured.

**NOTE**

To find your Agile Central API key, log in to the Agile Central Application manager, and select **API Keys**. Contrast saves the username, password and Agile Central URL entered in your configuration as a set of credentials.

4. Once you complete the fields, select **Test connection**. The test verifies that Contrast can reach the Agile Central instance and that the specified user can log in.
5. Once connected, select the **Applications** that you want to be available to this integration.
6. Choose a **Project name** and **Owner** from the dropdowns.
7. In the **Default priority** section, use the dropdowns to choose a priority level for each vulnerability severity.
8. Choose the **Environment** for which you want to generate tickets.
9. Choose a **Defect state**.
10. Add a name that the tickets are **Submitted by**.

**NOTE**

While none of these configuration fields are required, Agile Central may populate tickets with their own default values for any fields you leave blank.

11. To add another integration once you're connected in Contrast, select **Add Configuration** in the Agile Central row.
12. To automatically create tickets for newly discovered vulnerabilities, check the designated box in the configuration form. In the multiselect field that appears, choose the **Rules** and **Severity** levels of the vulnerabilities for which you want to generate tickets.

**NOTE**

This selection doesn't generate tickets retroactively.

## Manage Agile Central credentials

Contrast saves the latest set of credentials that you enter in your Agile Central configurations to help you set up new connections faster. The API key and URL values that you enter in your first configuration become the default credentials for your following configurations.

In subsequent configurations, Contrast will pre-populate the fields with the default credentials, but allow you to modify the values as needed. You can also manage your saved sets of credentials to simultaneously update all of the affected configurations.

To create or edit a configuration with credentials that are different than your default set:

1. Select the **Manage credentials** link.
2. In the **URL** field, use the dropdown to choose a set of saved credentials; or, manually update the values in the **URL**, **Username** and **Password** fields.
3. Once you've updated the fields, select **Test Connection**.
4. Select **Save**. If you're using new credentials, you must choose to override the existing set of credentials under the given name, or save the new values as a new credential set under a different name.

- To modify an existing set of saved credentials, select **Rename** if needed. Then select **Test Connection** and **Save**.

**NOTE**

Any updates to a set of credentials will affect all configurations using that set.

## Integrate with AWS Security Hub using Contrast Assess

Integrate Contrast Assess with AWS Security Hub to ensure a steady and secure transfer of security insights and findings directly to AWS, which helps in maintaining and enhancing your security posture through streamlined integration.

### Before you begin

Before you start you must have:

- AWS Account Number
- AWS Region
- The Contrast application from which to send insights

### Configure

There are two required steps to configure this integration:

- Configure AWS Security Hub to accept findings from Contrast
- Configure Contrast Assess to send findings to AWS Security Hub

### Configure AWS Security Hub to accept findings from Contrast

To allow AWS Security Hub to accept findings from Contrast:

1. Open the AWS Security Hub console associated with the AWS account and region where you want to receive findings from Contrast.
2. Go to the **Integrations** section and search for *Contrast Security*.
3. Locate the Contrast Security tile, click **Accept findings** and follow the subsequent prompts to complete the setup.
4. Continue with configuring Contrast Assess to send findings to AWS Security Hub.

### Configure Contrast Assess to send findings to AWS Security Hub

After configuring the AWS Security Hub, the next step is to configure Contrast Assess to send the findings to the AWS Security Hub:

1. In Contrast, go to the **user menu** and select **Organization settings > Integrations**.
2. Find and select the **AWS Security Hub** integration section.
3. Select **Manage Credentials**.

The screenshot shows the AWS Security Hub console interface. At the top, there's a header with the AWS logo and the text "AWS Security Hub" followed by a subtitle "Enables the Contrast Platform to publish Security Findings to AWS Security Hub." To the right of the header are two buttons: "Manage Credentials" and "Configure Applications". Below the header is a form with two input fields: "AWS Account Id" and "AWS Region". The "AWS Region" field is a dropdown menu currently showing "us-east-1". At the bottom right of the form are two buttons: "Save" and "Cancel".

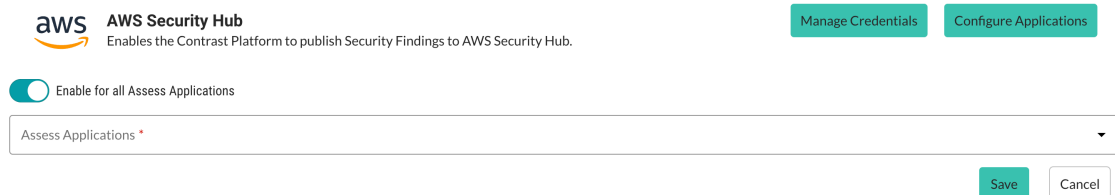
4. Enter the **AWS Account** number and select the **AWS Region**.
5. Select **Save**.

6. Continue by setting up applications in Contrast Assess.

## Set up applications in Contrast Assess

Once your credentials are set up, proceed to configure the applications:

1. In the **AWS Security Hub** integration section in Contrast, select **Configure Applications**.
2. Select whether to activate the AWS Security Hub integration for all Assess applications or select specific application names from a list, choosing which insights to forward.



The screenshot shows the 'AWS Security Hub' integration configuration page. At the top, there's a header with the AWS logo and the text 'AWS Security Hub' and 'Enables the Contrast Platform to publish Security Findings to AWS Security Hub.' To the right of the header are two buttons: 'Manage Credentials' and 'Configure Applications'. Below the header, there's a toggle switch labeled 'Enable for all Assess Applications'. Below the toggle is a dropdown menu labeled 'Assess Applications \*'. At the bottom right, there are 'Save' and 'Cancel' buttons.

3. Select **Save**.

## Retry mechanism

In case synchronization between Contrast Assess and AWS Security Hub fails, a retry mechanism ensures data reliability:

- If an event fails to sync, it will be stored and retried every night at midnight GMT.
- The retry count will increase by one with a maximum of three retries for up to 72 hours. After the third unsuccessful retry, the event will be discarded.
- If a vulnerability creation event fails and is stored, any subsequent update or delete action relating to that failed event will be stored and replayed in chronological order to maintain the correct state.

## Integrate with Amazon Security Lake using Contrast Assess

Integrate Contrast Assess with Amazon Security Lake to push findings and other relevant security data automatically.

### Before you begin

Before you start, you must have:

- AWS Region

## Create a Custom Source in AWS

1. In AWS, go to [Amazon Security Lake](#).
2. Select **Custom Source**.
3. Click **Create New Source**.
4. Enter a **Data source name** of your choice.
5. Set the **OCSF Event class** to *Security Finding*.
6. Input the **AWS account ID** and **External ID**.  
These IDs are under the *Amazon Security Lake* section on Contrast's **Integrations** page. From the user menu in Contrast select **Organization settings > Integrations**.
7. Once a custom source is created, retrieve the generated **Role ARN** and the **S3 ARN**. You will need these to connect to AWS.
8. Continue with connecting to Amazon Security Lake.

## Connect to Amazon Security Lake

1. In Contrast, go to the user menu and select **Organization settings > Integrations**.

- Find and select the **Amazon Security Lake** integration section.
- Select **Manage Credentials**.

The screenshot shows the 'Amazon Security Lake' configuration interface. At the top, it says 'Amazon Security Lake' and 'Enables the Contrast Platform to send OCSF Security Finding data to Amazon Security Lake on your behalf.' Below this, there are two tabs: 'Credentials' (selected) and 'Applications'. Under 'Credentials', there are two input fields: 'External ID' with the value '35346c29-adc8-4be1-831f-bcf72bf86662' and a 'Rotate' button, and 'AWS Account Id' with the value '763284681916'. Below these are three more fields: 'Role ARN', 'S3 ARN', and 'AWS Region' (set to 'us-east-1'). At the bottom right, there are 'Save' and 'Cancel' buttons.

- Enter the generated **Role ARN** and the **S3 ARN** from before.
- Choose the **AWS Region** from the list or enter it manually.
- Select **Save**.
- Continue by setting up applications in Contrast Assess.

## Set up applications in Contrast Assess

Once your credentials are set up, proceed to configure the applications:

- In the **Amazon Security Lake** integration section in Contrast, select **Applications**.
- Select whether to activate the Amazon Security Lake integration for all Assess applications or select specific application names from a list.
- Select **Save**.

## Retry mechanism

In case synchronization between Contrast Assess and Amazon Security Lake fails, a retry mechanism ensures data reliability:

- If an event fails to sync, it will be stored and retried every night at midnight GMT.
- The retry count will increase by one with a maximum of three retries for up to 72 hours. After the third unsuccessful retry, the event will be discarded.
- Suppose a vulnerability creation event fails and is stored. Any subsequent update or delete action relating to that failed event will be stored and replayed in chronological order to maintain the correct state.

## Integrate with Azure Boards

With an Azure Boards integration with Contrast, you can: automatically generate tickets for bugtracking, synchronize comments and push notifications for your applications.

- Automatically generate tickets or work items for bugtracking, synchronization of comments, and push notifications for your applications.  
The integration sends session metadata that you configured in the agent configuration file to Azure Boards work items.
- Configure a two-way integration
- Exclude sensitive information that Contrast sends to Azure Boards

## Azure Boards requirements

- Account credentials for Azure Boards or TFS: username and personal access token (PAT).
- Scope to read and write work items with your PAT.

- An Azure Boards or TFS instance, accessible by HTTP to Contrast.
- An instrumented application in Contrast that is also associated to an Azure Boards project.

## See also

- Microsoft's [Azure Boards documentation](#).
- [Connect with Azure Boards \(page 1066\)](#)
- [Automatically create tickets \(page 1067\)](#)
- [Two-way integration \(page 1067\)](#)
- [Exclude sensitive information \(page 1068\)](#)
- [Set personal access tokens \(page 1069\)](#)

## Connect to Azure Boards

### Steps

1. In Contrast, go to **Organization settings > Integrations**.
2. For the Azure Boards integration, select **Connect**.
3. Enter the following values:
  - **Name:** Label that will display when Contrast sends findings to bugtrackers in Azure Boards.
  - **URL:** Azure Boards or TFS URL. Contrast must be able to access this.
  - **Version:** Contrast uses API v2 to support Azure DevOps Services, TFS 2015 and TFS 2017.
  - **Personal access token:** An alternate password to authenticate to your host.
4. Select **Test connection**. This may take a few minutes, depending on the number of Azure Boards or TFS projects. The test verifies that Contrast can reach the Azure Boards or TFS instance you entered, and it accepts the user's PAT to login.
5. Once Azure Boards is connected, select the Contrast **Applications** you want to make available to this bugtracker.
6. Enter values for **Project**, **Assignee** and **Work Item Type**.
7. Select a **Team**, then select an **Area** within the team. This will send tickets to a specific backlog.
8. Set the **Default priority** for vulnerability severity levels. This prioritizes tickets to fix vulnerabilities for the selected applications, based on severity. At this point, Contrast will make an API call and return a list of Azure Boards or TFS ticket states.
9. You can also set up the following options for Azure Boards:
  - [Two-way integration \(page 1067\)](#) to automatically update vulnerability status in Contrast
  - [Automatic ticket creation \(page 1067\)](#)
  - Exclude sensitive information from issue titles and additional fields when creating tickets
  - Send tags as labels when creating tickets

## See also

- [Set personal access tokens \(page 1069\)](#)
- [Manage credentials \(page 1066\)](#)

## Manage Azure Boards credentials

Contrast saves the most recent credentials for an Azure Boards integration to help you set up new connections faster. The values for the Azure Boards URL and version that you enter in your first configuration are the default credentials for the next Azure Boards integration. Contrast pre-populates the next Azure Boards configuration with these default credentials, but you can modify or delete them.

## Change Azure Boards credentials

1. Go to the **user menu > Organization Settings > Integrations**.
2. Under Azure Boards, select **Manage Credentials**.
3. From the dropdown in the Credentials box, select the credentials you want to change.
4. If you want to rename the credentials, select **Rename**.
5. If you want to change the URL, enter a new URL.
6. Select a version.
7. Enter an Azure Personal Access Token.
8. Select **Test connection** to be sure the changes work.
9. Select **Save**.

## Delete Azure Boards credentials

1. Go to the **user menu > Organization Settings > Integrations**.
2. Under Azure Boards, select **Manage Credentials**.
3. From the dropdown in the Credentials box, select the credentials you want to change or delete.
4. Select **Delete credential** at the bottom of the page.
5. When prompted to confirm the deletion, select **Continue** in the Delete credential window.

## See also

- [Connect with Azure Boards \(page 1066\)](#)
- [Set Azure Boards personal access tokens \(page 1069\)](#)

## Automatically create tickets

You can **automatically create tickets** every time Contrast discovers new vulnerabilities.

## Steps

1. In the [Azure Boards integration panel \(page 1065\)](#), select **Automatically create tickets for new vulnerabilities discovered**. This displays a multi-select field for **Rules** and **Severity**.
2. Select the rules or severity levels of vulnerabilities that should trigger a new ticket in Azure Boards or TFS. **Critical** and **High** are the default selections.



### NOTE

This setting only works for new vulnerabilities discovered after you select it.

## See also

- [Connect with Azure Boards \(page 1066\)](#)
- [Two-way integration \(page 1067\)](#)
- [Set personal access tokens \(page 1069\)](#)

## Two-way integration

You can use a two-way integration with Azure Boards. This will automatically update the status of a vulnerability in Contrast when you close or reopen an issue in Azure Boards or TFS that links to the vulnerability .

## Steps

1. In the [Azure Boards integration panel \(page 1065\)](#), select **Enable two-way integration**. This displays Vulnerability Status fields.
2. Select the dropdowns to set a vulnerability status for each Azure Boards or TFS ticket state.
3. **Save** the two-way integration. Contrast will populate the vulnerability status in Azure Boards or TFS tickets.
4. When you update the state of a ticket in Azure Boards or TFS, Contrast will automatically generate comments in the **Discussion** tab for that vulnerability. Each comment includes the name of the bugtracker and a link to the ticket.



### NOTE

If you select the vulnerability status **Not a problem** as a ticket state in Azure Boards or TFS, Contrast also requires you to select a **Reason**. The default value is **Other**.



### CAUTION

For multiple vulnerabilities sent to Azure Boards or TFS as a single issue, the ticket state applies to all vulnerabilities associated with that ticket. Conversely, when you link multiple tickets to a single vulnerability, you must update all associated tickets before you can update the vulnerability. For example, if you change a ticket state from **New** to **Active**, Contrast updates the vulnerability status only if all tickets related to that vulnerability also have an **Active** state.

## See also

- [Set personal access tokens \(page 1069\)](#)
- [Connect to Azure Boards \(page 1066\)](#)
- [Automatically create tickets \(page 1067\)](#)

## Exclude sensitive information with Azure Boards

You can set up a Contrast Azure Boards configuration to mask sensitive information that it sends to tickets or work items in Azure Boards.

## Steps

1. Under Organization settings, select **Integrations**.
2. Under Azure Boards, select **Show configurations** and select the configuration you want to update.
3. In the [Azure Boards integration panel \(page 1065\)](#), select **Allow exclusion of sensitive information from issue titles and additional fields when creating tickets**.
4. **Save** the configuration.

## See also

- [Connect to Azure Boards \(page 1066\)](#)
- [Automatically create tickets \(page 1067\)](#)



## Set Azure Boards personal access tokens

Personal Access Tokens (PAT's) are used by Contrast to access Azure Boards. PAT's can be set to provide full access or "a la carte" access.

### Steps

1. Navigate to **User Settings** in Azure.
2. Select **Personal access tokens**. Existing personal access tokens are displayed.
3. Click **New Token**. The **Create a new personal access token** modal displays.
4. Enter required fields and scopes for your new PAT in the **Create a new personal access token** modal.



#### NOTE

**Read, write, & manage** under **Work items** must be selected to work with Contrast.

##### Work Items

Work items, queries, backlogs, plans, and metadata



Read



Read & write



Read, write, & manage

5. Click **Create** to finish and save your new personal access token.

### See also

- [Connect with Azure Boards \(page 1066\)](#)
- [Automatically create tickets \(page 1067\)](#)
- [Two-way integration \(page 1067\)](#)

## Azure Pipelines extension

Use the Azure Pipeline extension to integrate Contrast with your deployment workflow. The following instructions guide you through the steps to set up and configure the extension for your Contrast instance.

Before you begin to set up the extension, make sure that you have the privileges to install a Microsoft extension. If not, you can [request the extension](#) for a project.

### Install and configure the Azure Pipelines extension

Use these procedures to install the extensions and create a service connection.

#### Install the extension

Follow the [Microsoft instructions](#) to install the extension **Contrast ADO Pipeline Integration**.

#### Create a service connection

The Contrast Server Connection service connection lets Azure DevOps integrate with Contrast.

1. Go to your **Project Settings** at the bottom of the sidebar. You'll need to be part of the Project administration group or have enough permissions to alter the settings.
2. In the **Pipelines** section of the settings menu, select **Service connections**.

3. Select **New Service connection** and then **Contrast Server Connection**.
4. Complete all the fields with required data:
  - **Contrast URL:** The URL for your Contrast instance. Exclude `/Contrast` at the end; only the host name is required.
  - **Contrast credentials:** Copy these credentials from your [personal keys \(page 598\)](#) under the **user menu > User settings** in the Contrast web interface:
    - Organization UUID (Organization ID)
    - Service Key
    - API Key
  - **Contrast username:** The user name you use to log in to Contrast.
5. Optionally, select the **Logging** option.  
If enabled, logs are stored in the pipeline artifact, providing visibility into the task execution and Contrast API interactions.

## Configure a task in the Azure Pipelines extension

Use this procedure to configure a release or build pipeline task.

### Steps

1. Select the pipeline where you want to add the task then select **Edit**.
2. For release pipelines, select a stage for which you want to add the task.
3. To add the task, click the ellipsis (...) menu and select **Add an agentless job**.
4. Select the **+** button next to your agentless job, and add the **Contrast Assess Security** task.
5. To choose a connection and application, select a **Service Connection** from the **Contrast Service Connection** menu. You can also select **Manage** to go to the **Service connections settings** in your **Project Settings**.
6. Optionally, select a different vulnerability source. The default selection is **Assess**.  
If you select **Assess-Libraries** as the Vulnerability source, Contrast looks at the number of instances where that library is used according to the specified severity criteria, not the number of libraries. For example, you could have 11 libraries with 45 vulnerable instances of those libraries.
7. Select one of your applications from the **Application** menu.
8. To configure the task, use the **Allowed Status** and **Build Number** fields to filter your results from Contrast. Leave them blank if you don't want to filter results. The values set in these fields will be validated against the conditions you configure in the following fields.
9. Proceed to your severity counters, where you must set the maximum number of vulnerabilities allowed per severity. If your selected application has more vulnerabilities than allowed for that severity level, your task will fail.

For build pipelines only: If you want to prevent the execution of a job if the task fails, you must set the job to depend on the agentless job that includes the Contrast task.

1. Select the job you want to prevent from executing.
2. In the **Dependencies** section, add the **Agentless job**.



#### NOTE

You can only use this task for an agentless job.

## Add a release gate to a pipeline in Azure Pipelines

Release gates offer a safeguard to prevent deployment to environments if vulnerabilities for a given application exceed a certain threshold.

### Before you begin

- Microsoft Documentation offers more information on how to [define a gate for a stage](#) and [how to configure a gate](#).
- Microsoft Documentation offers more information on how to [define a gate for a stage](#) and [how to configure a gate](#).
- Optionally, create a Contrast Security service connection.

### Add and configure a gate

1. Find the release pipeline where you want to add the gate and select **Edit**.
2. Choose the stage and deployment conditions for the gate. They can either be pre-conditions or post-conditions. You can add multiple gates to the same conditions.
3. Under **Gates**, enable the gate you created.
4. Select **Add** and then **Verify application vulnerabilities**.
5. Select your Contrast service connection.  
If you haven't already created a service connection, select **New** next to the service connection dropdown to create the Contrast service connection. Fill in all the fields and select **OK**.  
Select **Refresh list**, then select your newly created connection.
6. Hover on the field or select **Refresh** to see a list of applications. Select the one that is most appropriate to the release pipeline.
7. If you want, you can select which vulnerability status or build numbers will be used for filtering when retrieving the data for the gate evaluation.
8. Set the maximum amount of vulnerabilities allowed per severity or total threshold based on a threshold definition of Split or Combined.  
If any validations fail when your pipeline reaches this gate, the pipeline will keep requesting samples until it becomes valid, or until the evaluation times out.



#### TIP

- You can customize **Evaluation** options to configure the time between the re-evaluation of gates. For instance, you can set this value to 24 hours so that the gates will evaluate every day. This way you can remediate vulnerabilities and pass the required gate conditions without having to re-initiate the execution of the pipeline from start (or obtain manual approvals if they exist).
- **Split threshold definition:** If you set the threshold definition to **Split**, specify the maximum allowed vulnerabilities for each severity level. The evaluation compares each severity value against its defined threshold.
- **Combined threshold definition:** If you set the threshold definition to **Combined**, specify the total allowed threshold for the sum of all severity levels. The evaluation compares the combined severity count against the defined total threshold.

## Find logs for Azure Pipelines

When you turn on logging, pipeline task details are captured and stored as log files in the pipeline artifact.

## Before you begin

Verify that the **Logging** option in the [service connection \(page 1069\)](#) is selected.

## Steps

1. Run the pipeline task.  
The log file with the task results is stored as a pipeline artifact.
2. **Build pipelines:** Locate the log file:
  - a. Go to **Azure DevOps** → **Pipelines** → **Runs**.
  - b. Select a pipeline run.
  - c. Go to **Artifacts**.
  - d. Select **Logs**.
  - e. Download the log file.
3. **Release pipelines:** Locate the log file:
  - a. Go to **Azure DevOps** → **Pipelines** → **Releases**.
  - b. Select a release.
  - c. Hover on the stage and select **Logs**.
  - d. Select **Download all logs**.

## Use Azure Service Fabric with the .NET Framework or .NET Core agent

If you are using a container image, follow the instructions to [install in containers \(page 220\)](#). Otherwise, to add the Contrast .NET Framework or .NET Core agent to an Azure Service Fabric service:



### TIP

For Standalone Executable services, the `ServiceManifest.xml` file is located in the top-level Azure Service Fabric project (for example, the `sfproj` file).

1. Install the appropriate NuGet package to the main project for the service.
  - **.NET Framework:** Install `Contrast.NET.Azure.AppService`. All files in the `contrastsecurity` folder must have `Copy to Output Directory` set to `Copy if newer`.
  - **.NET Core:** Install `Contrast.SensorsNetCore`. All files in the `contrast` folder have `Copy to Output Directory` set to `Copy if newer`.
2. Set `ServiceManifest/CodePackage/EntryPoint/ExeHost/WorkingDirectory` in `ServiceManifest.xml` to `CodePackage`.

```
<CodePackage Name="Code" Version="1.0.0">
 <EntryPoint>
 <ExeHost>
 <Program>DemoNetFxStatelessService.exe</Program>
 <WorkingFolder>CodePackage</WorkingFolder>
 </ExeHost>
 </EntryPoint>
</CodePackage>
```

3. Set environment variables in `ServiceManifest.xml` to configure the profiler.
  - **.NET Framework:**

```
<CodePackage>
 <EnvironmentVariables>
 <EnvironmentVariable Name="COR_ENABLE_PROFILING">
```

```
Value="1" />
 <EnvironmentVariable Name="COR_PROFILER"
Value="{EFEB8EE0-6D39-4347-A5FE-4D0C88BC5BC1}" />
 <EnvironmentVariable
Name="COR_PROFILER_PATH_32" Value=".\contrastsecurity\runtimes\win-
x86\native\ContrastProfiler.dll" />
 <EnvironmentVariable
Name="COR_PROFILER_PATH_64" Value=".\contrastsecurity\runtimes\win-
x64\native\ContrastProfiler.dll" />
 <EnvironmentVariable Name="CONTRAST_CONFIG_PATH"
Value="contrast_security.yaml" />
```

- **.NET Core:**

```
<CodePackage>
 <EnvironmentVariables>
 <EnvironmentVariable Name="CORECLR_ENABLE_PROFILING"
Value="1" />
 <EnvironmentVariable Name="CORECLR_PROFILER"
Value="{8B2CE134-0948-48CA-A4B2-80DDAD9F5791}" />
 <EnvironmentVariable Name="CORECLR_PROFILER_PATH_32"
Value="contrast\runtimes\win-x86\native\ContrastProfiler.dll" />
 <EnvironmentVariable Name="CORECLR_PROFILER_PATH_64"
Value="contrast\runtimes\win-x64\native\ContrastProfiler.dll" />
 <EnvironmentVariable Name="CONTRAST_CONFIG_PATH"
Value="contrast_security.yaml" />
```

4. Configure the agent with either:

- **A YAML file:** Add it to the main project for the service. Make sure Copy to Output Directory for the file is set to Copy if newer. Add an environment variable to ServiceManifest.xml specifying the location of the file, like this:

```
<CodePackage>
 <EnvironmentVariables>
 <EnvironmentVariable Name="CONTRAST_CONFIG_PATH"
Value="contrast_security.yaml" />
```

- **Environment variables:** Add them to ServiceManifest.xml, like this:

```
<CodePackage>
 <EnvironmentVariables>
 <EnvironmentVariable Name="CONTRAST__API__URL"
Value="https://teamserver-staging.contsec.com" />
 <EnvironmentVariable Name="CONTRAST__API__API_KEY"
Value="aBcD0123" />
 <EnvironmentVariable Name="CONTRAST__API__SERVICE_KEY"
Value="ABCD0123" />
 <EnvironmentVariable Name="CONTRAST__API__USER_NAME"
Value="agent_123@Team" />
```

5. Deploy the Azure Service Fabric application as usual.

## Contrast - Bamboo plugin

This plugin adds functionality to Bamboo so that you can configure profiles for connecting to Contrast and verify builds against vulnerability thresholds.

### Install and configure

To install and configure the Bamboo plugin:

1. Download the Contrast Bamboo plugin ( *contrast-bamboo-plugin-#.#.#.jar*) from the [Bamboo Marketplace](#).
2. Select **Add-Ons** from the top-left settings menu.
3. Select **Upload add-on**.
4. When prompted to upload a file, select *contrast-bamboo-plugin-#.#.#-SNAPSHOT.jar*.
5. Verify you see the plugin under **User-installed add-ons**.
6. Now that the plugin is installed, configure a profile for Contrast. Select **Contrast Profiles** under **Add-Ons** in the side navigation bar.
7. In the **Profile Configuration** page, select **New Profile** and complete the form.

**NOTE**

If you are a hosted customer, you do not need to enter a Contrast URL.

8. Select **Test Connection** to verify that your settings are correct. A success notification will appear when a connection is established.

## Configure vulnerability thresholds

The Bamboo plugin can be added as a task to build jobs to check for vulnerability conditions that you configure. This checks Contrast for the number and type of vulnerabilities in the applications.

To add a task to a build job:

1. Select **Create a New Build Plan** (you can also use an existing plan).
2. Enter a project name, plan name and link to the repository host. The project key and plan key is auto-generated.
3. Once you create the plan, add a task to the build process by selecting **Add Task**.
4. In the window that appears, find the **Contrast CI for Assess** task and select it.  
The **Tasks** configuration page relies on a Contrast profile, as well as a server name, application name and a **Passive** parameter. The server name isn't required, but should correspond to a server name in Contrast if used. The application name must be on the designated server.  
If you select the **Passive** parameter, the plugin will query all vulnerabilities for the application (not just build-specific vulnerabilities). If you do this, you don't have to run the application with its integration tests before the Contrast post-build action in the Bamboo build.
5. Next, define conditions for when to fail a build:
  - **Threshold Count:** The minimum number of findings required to fail the build.
  - **Threshold Severity:** The minimum severity at which to count a finding towards the threshold count.
  - **Threshold Vulnerability Type:** The type of finding required to count a finding towards a threshold count.

**NOTE**

Using the **Any** option means that any severity or vulnerability type is counted towards the maximum threshold count.

6. Select **Add New Threshold Condition** to configure multiple conditions for each task.
7. Select **Save**.
8. Enable the build plan by selecting the checkbox in the bottom left.

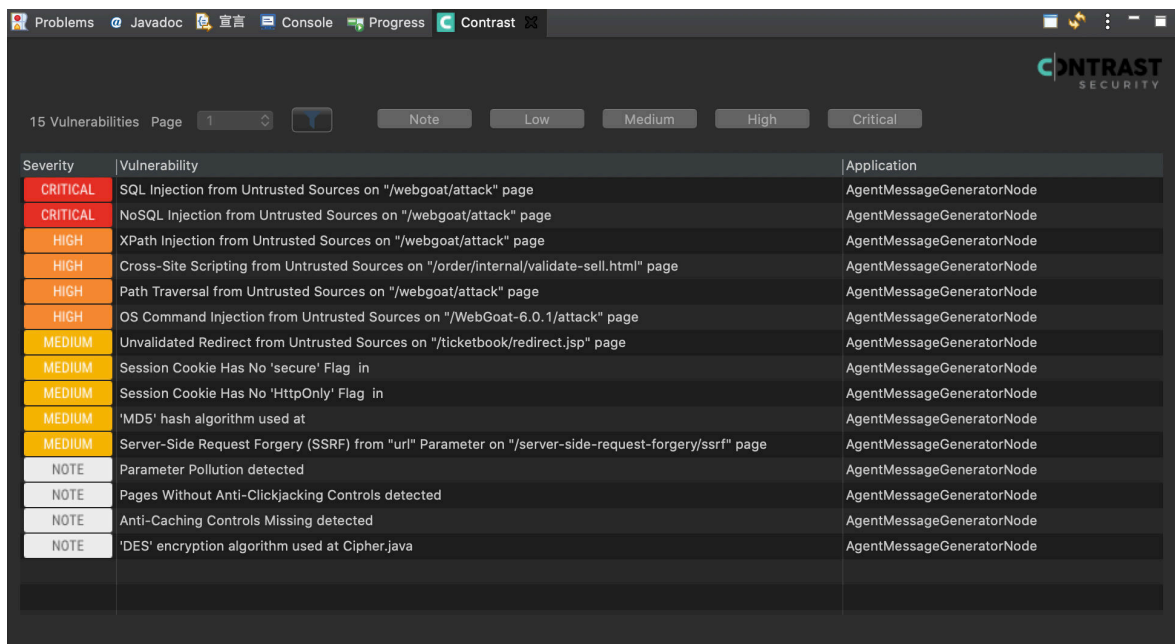
## Eclipse

For applications instrumented with a Contrast agent, you can view vulnerability information directly in the Eclipse IDE during development for faster remediation. You will see affected lines of code and can view more details about the vulnerability in Contrast.

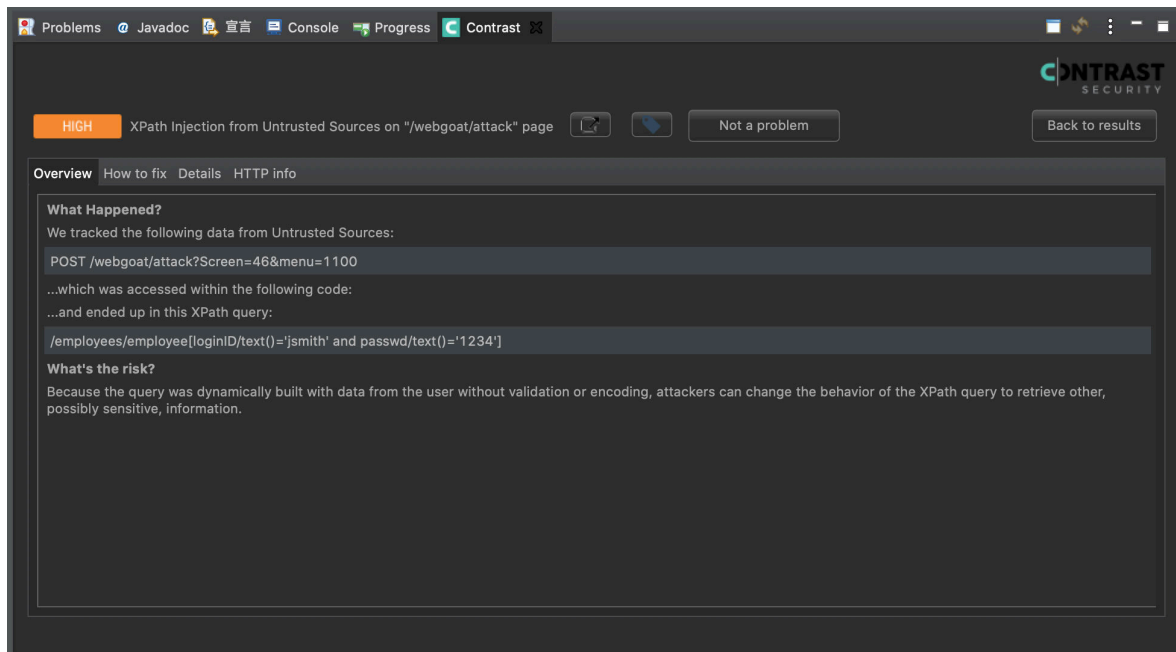
This plugin is available for Mac/OS, Linux, and Windows and supports the latest versions of Eclipse.

To install the Eclipse plugin, visit the [Eclipse marketplace](#) or:

1. In Eclipse, select **Help > Eclipse Marketplace**.
2. Search “Contrast Security”.
3. Select **Install**.
4. Configure the plugin at **Window > Show View > Other**.
5. Search “Contrast” and add the view that appears in the search.
6. Enter the **Username**, **API Key**, **Organization ID** and **Service Key** in the configuration page. You can [find these keys \(page 598\)](#) in user settings.
7. Select **Add**.
8. The **Vulnerabilities** view shows a list of all the vulnerabilities from Contrast. You can sort and filter them.



9. Select the vulnerability title for information about that particular vulnerability. From there you can select **How to fix** for remediation instructions to fix the vulnerability. Select **Details** and double-click on the Java stack traces to focus on a particular source code line. You can also change the vulnerability status here.



10. Select the **Go to page** icon to open Contrast and see more vulnerability information.

## Generic webhooks

Contrast supports a generic webhook integration to receive [notifications \(page 1175\)](#) on any URL that receives POST messages. You can add [custom variables \(page 1077\)](#) to your payload like `$ApplicationName` and `$ServerId` when a [Contrast event \(page 1080\)](#) triggers them.

A generic webhook stays connected as long as the receiving end returns any 2XX HTTP response code. If the generic webhook receives too many non-2XX response codes, it disconnects.

### Connect

To connect a generic webhook:

1. Retrieve the URL from which you want Contrast to receive notifications.
2. In the user menu, select **Organization settings > Integrations**.
3. In the **Generic webhook** integration option, select **Connect**.
4. Name the webhook, and paste the URL in the designated field.
5. Select the application(s) that you want to filter.
6. In the **Payload** field, enter a [variable \(page 1077\)](#). For example:

```
{
 "title": "$Title",
 "message": "$Message"
}
```

7. Select **Add**.

To test the webhook:

1. Go to **Organization settings > Notifications**.
2. In the dropdown under **Integrations**, select the generic webhook name.
3. For each **Subscription** (event type) you want to be notified of, click the toggle in the **Integrations** column.
4. Cause an event type to occur, and confirm that you get a notification at the URL specified.




**NOTE**


If this webhook fails to return a successful response after 5 attempts, it will be disconnected. To restore the configuration, you must retest the connection and resave it.

You can configure the integration so that all Organization Administrators are notified in the Contrast application and by email when Contrast disconnects a generic webhook.


To do this, go to the same location: **Organization settings > Integrations > Generic webhook > Show configurations**. Select the name of the connection you want to configure. Then select the **Notify on disconnect** checkbox to receive notifications and click **Save**.

Connect with  webhook

Name

URL 

Applications

Payload (Optional) 

☒ Send as HTML

☒ Notify on disconnect  
If this webhook fails to return a successful response after 5 attempts, it will be disconnected. All organization administrators will be notified in Contrast and by email.

Cancel Test URL Add

**Generic webhook variables**

You can customize your [generic webhook \(page 1076\)](#) response with data from Contrast events such as `NEW_VULNERABILITY` and `SERVER_OFFLINE`. Each event contains variables you can call in your payload request. Variables are either for general use or for an application, server or vulnerability.

Variables	Description
<b>General variables</b>	
\$EventType	The event type responsible for triggering the webhook For example: SERVER_OFFLINE
\$Message	A message summarizing the event that triggered the webhook
\$OrganizationId	The unique ID Contrast assigns to an organization when it is created
\$OrganizationName	The name of your organization
\$Title	Always returns "Contrast Security"
<b>Application variables</b>	
\$ApplicationChild	Returns true if the application is a child application, false if not
\$ApplicationCode	A secondary shorthand that appears in the title of an application, and is blank by default For example: TEST
\$ApplicationContextPath	The context path of the application For example: /example/somethingelse
\$ApplicationFirstSeen	When the application was first seen, in Unix time For example: 1572033840000
\$ApplicationHasParentApp	Returns true if the application has a parent, false if not
\$ApplicationImportance	Enumerated value of the application Importance level For example: MEDIUM
\$ApplicationId	The unique ID Contrast assigns to an application when it is created For example: 49fe2978-1833-4441-83db-2b7o486d9413
\$ApplicationImportanceDescription	The importance level assigned to the application For example: Medium
\$ApplicationLanguage	The programming language of the application
\$ApplicationLastSeen	When the application was last seen, in Unix time For example: 1572033840000
\$ApplicationLicenseLevel	Whether or not the application has an Assess license Values: Licensed, Unlicensed
\$ApplicationMaster	Returns true if the application is a primary application, false if not
\$ApplicationName	The name of the application
\$ApplicationParentAppId	The unique ID Contrast assigns to an application when it's created, in this case, the parent application, if it exists For example: 49fe2978-1833-4441-83db-2b7o486d9413
\$ApplicationTags	A comma separated list of the Application tags.
\$ApplicationTotalModules	The number of modules your application has
<b>Server variables</b>	
\$Environment	The environment of the server For example: DEVELOPMENT or PRODUCTION
\$ServerId	The ID of the server involved in the event If more than one server is involved, this is a comma-delimited list of server IDs.
\$ServerName	The name of the server involved in the event If more than one server is involved, this is a comma-delimited list of server names
<b>Vulnerability variables</b>	
\$Severity	If this event is triggered by a vulnerability, this is the severity of the vulnerability
\$Status	If this event is triggered by a vulnerability, this is the status of the vulnerability

Variables	Description
\$TraceId	If this event is triggered by a vulnerability, this is the vulnerability ID
\$VulnerabilityAgentLanguage	The application language or framework name of the where the vulnerability was discovered (for example,.Java, .NET, Ruby, and so forth.)
\$VulnerabilityAppVersionTags	The application versions the vulnerability is found in For example: v1.2.3
\$VulnerabilityAutoRemediatedExpirationPeriod	Auto-remediated expiration period for the vulnerability, in Unix time For example: 1572033840000
\$VulnerabilityBugTrackerTickets	A comma delimited list of tickets created when the vulnerability was sent to bugtracker For example: ticket1, ticket2, ticket3
\$VulnerabilityCategory	The category of vulnerability found For example: Injection
\$VulnerabilityClosedTime	When the vulnerability was closed, in Unix time For example: 1572033840000
\$VulnerabilityConfidence	Confidence of the vulnerability
\$VulnerabilityDefaultSeverity	Default severity of the vulnerability
\$VulnerabilityDiscovered	When the vulnerability was first discovered, in Unix time For example: 1572033840000
\$VulnerabilityEvidence	The evidence of the vulnerability
\$VulnerabilityInstanceUuid	The unique ID Contrast assigns to a vulnerability instance when it is created For example: R33T-N00B-TGIF-RM6P
\$VulnerabilityFirstTimeSeen	When the vulnerability was first seen, in Unix time For example: 1572033840000
\$VulnerabilityImpact	The impact level of the vulnerability Values: Low, Medium, High
\$VulnerabilityLastTimeSeen	Last time the vulnerability was seen, in Unix time For example: 1572033840000
\$VulnerabilityInstanceLastTimeSeen	Last time the vulnerability was seen, in Unix time For example: 1572033840000
\$VulnerabilityLicenseLevel	License level of the vulnerability
\$VulnerabilityLikelihood	The likelihood of the vulnerability Values: Low, Medium, High
\$VulnerabilityReportedToBugTracker	When the vulnerability was sent to a bugtracker, in Unix time For example: 1572033840000
\$VulnerabilityReportedToBugTrackerTime	Returns true If the vulnerability was sent to a bugtracker
\$VulnerabilityRule	Rule associated with the vulnerability
\$VulnerabilityRuleName	Name of the rule associated to the vulnerability
\$VulnerabilityRuleTitle	Title of the rule associated to the vulnerability
\$VulnerabilitySubStatus	Substatus of the vulnerability
\$VulnerabilityTags	Custom tags associated with the vulnerability For example: my-custom-tag
\$VulnerabilityTitle	Title of the vulnerability
\$VulnerabilitySubStatusKeyCode	Key code of the vulnerability substatus
\$VulnerabilityTotalTracesReceived	Total number of times the vulnerability was received
\$VulnerabilityUuid	The unique ID used to look up a vulnerability
\$VulnerabilityVisible	true if the vulnerability is licensed and visible, false if not
\$VulnerabilityRule	If event is triggered by a vulnerability, this is the rule that the vulnerability violated
\$VulnerabilityTags	If event is triggered by a vulnerability, this is a comma-delimited list of tags associated with the vulnerability

## Events and generic webhook variables

You can customize your [generic webhook \(page 1076\)](#) response with data from Contrast events such as `NEW_VULNERABILITY` and `SERVER_OFFLINE`. Each event contains [general \(page 1078\)](#), [application \(page 1078\)](#), [server \(page 1078\)](#) or [vulnerability \(page 1078\)](#) variables you can call in your payload request.

Event	Variables
<code>ATTACK_END</code>	<a href="#">General (page 1078)</a> , <a href="#">Application (page 1077)</a> , <a href="#">Server (page 1078)</a>
<code>ATTACK_EVENT_COMMENT</code>	<a href="#">General (page 1078)</a> , <a href="#">Application (page 1078)</a> , <a href="#">Server (page 1078)</a>
<code>ATTACK_UPDATE</code>	<a href="#">General (page 1078)</a> , <a href="#">Application (page 1078)</a> , <a href="#">Server (page 1078)</a>
<code>EXPIRING_LICENSE</code>	<a href="#">General (page 1078)</a> , <a href="#">Application (page 1078)</a>
<code>NEW_ASSET</code> (if new application)	<a href="#">General (page 1078)</a> , <a href="#">Application (page 1078)</a> and <a href="#">Server (page 1078)</a> (if new application)
<code>NEW_ATTACK_APPLICATION</code>	<a href="#">General (page 1078)</a> , <a href="#">Application (page 1078)</a> , <a href="#">Server (page 1078)</a>
<code>NEW_ATTACK_UPDATE</code>	<a href="#">General (page 1078)</a> , <a href="#">Application (page 1078)</a> , <a href="#">Server (page 1078)</a>
<code>NEW_ATTACK</code>	<a href="#">General (page 1078)</a> , <a href="#">Application (page 1078)</a> , <a href="#">Server (page 1078)</a>
<code>NEW_VULNERABILITY_COMMENT</code>	<a href="#">General (page 1078)</a> , <a href="#">Application (page 1078)</a> , <a href="#">Server (page 1078)</a> , <a href="#">Vulnerability (page 1078)</a>
<code>NEW_VULNERABILITY</code>	<a href="#">General (page 1078)</a> , <a href="#">Application (page 1078)</a> , <a href="#">Server (page 1078)</a> , <a href="#">Vulnerability (page 1078)</a>
<code>NEW_VULNERABLE_LIBRARY</code>	<a href="#">General (page 1077)</a> , <a href="#">Application (page 1078)</a>
<code>SERVER_OFFLINE</code>	<a href="#">General (page 1078)</a> , <a href="#">Server (page 1078)</a>
<code>VULNERABILITY_CHANGESTATUS_CLOSED</code>	<a href="#">General (page 1078)</a> , <a href="#">Application (page 1078)</a> , <a href="#">Server (page 1078)</a> , <a href="#">Vulnerability (page 1078)</a>
<code>VULNERABILITY_CHANGESTATUS_OPEN</code>	<a href="#">General (page 1078)</a> , <a href="#">Application (page 1078)</a> , <a href="#">Server (page 1078)</a> , <a href="#">Vulnerability (page 1078)</a>
<code>VULNERABILITY_DUPLICATE</code>	<a href="#">General (page 1078)</a> , <a href="#">Application (page 1078)</a> , <a href="#">Server (page 1078)</a> , <a href="#">Vulnerability (page 1078)</a>

## Integrate with GitHub

Set up an integration to automatically send issues to GitHub when Contrast finds them in your applications.

Before you start, be sure you have:

- GitHub account credentials (username and personal access token). When you [generate your personal access token](#), be sure to enable the **repo** permissions.
- Access to a GitHub organization and repository for the application.
- Write permission ([push access](#)) to the repository. This is required to set labels, milestones and assignees in the configuration form.
- A running GitHub instance accessible via HTTP to Contrast.

To connect with GitHub:

1. Go to **Organization settings > Integrations** in the **user menu**.
2. Click **Connect** in the row for GitHub.
3. In the **Connect with GitHub** form, add the name for the bugtracker entry, the username for the account connected to GitHub and the password for the specified username in the appropriate fields. The GitHub URL must be accessible from the Contrast instance being configured.
4. Automatically create issues in GitHub for newly discovered vulnerabilities by checking the box at the bottom of the configuration form. In the multiselect field that appears, choose the **Rules** and **Severity** levels of the vulnerabilities for which you want to generate tickets. The default selections are **Critical** and **High**.

- Once you complete the fields, select **Test connection**. This process may take a few moments depending on the number of your GitHub organizations and repositories. The test verifies that the GitHub instance can be reached by Contrast and that the specified user is able to log in.
- Once a connection is made, select the **Applications** that you want to be available to this bugtracker.
- Select the values for the **GitHub organization** and **Repository** fields using the dropdowns.

**NOTE**

If you change the **GitHub organization** or **Repository** values, you must re-enter the values for optional fields.

- Optionally, add **Labels**, **Assignees** and a **Milestone** for GitHub issues using the given fields.

**NOTE**

For multiple vulnerabilities sent in bulk to GitHub as a single issue, the GitHub ticket status applies to all vulnerabilities associated with that ticket. For multiple issues tied to a single vulnerability, the vulnerability can only be closed when all the tickets are closed.

**See also**

[Integrate Contrast with GitHub and GitHub Advanced Security \(page 1081\)](#)

[Integration example: Assess and GitHub \(page 1083\)](#)

[GitHub actions \(page 1084\)](#)

**Integrate Contrast with GitHub and GitHub Advanced Security**

Contrast supports multiple methods of integrating GitHub source code management (SCM) with Contrast technologies.

For an example of how to integrate Assess with a GitHub workflow, visit [Assess and GitHub. \(page 1083\)](#)

**Integrate Contrast Scan with GitHub**

I want to use GitHub to...	Procedure	Related links
Scan code in my master branch	<p>Use the <a href="#">Contrast Scan local engine (page 672)</a> as part of your workflow.</p> <p>You have a choice of using the Scan CLI or the Contrast Scan Analyze GitHub action for the scan.</p> <p>The CLI does not support specific branch scanning. It creates separate projects for each repository that you scan. For this reason, using the Contrast Scan Analyze Github action is the recommended option.</p>	<p><a href="#">Scan CLI (page 1009)</a></p> <p><a href="#">Contrast Scan Analyze</a></p> <p><a href="#">Contrast Scan Analyze README</a></p>
Scan code in my personal or an alternative branch	<p>Use the Contrast Scan Analyze GitHub action for the scan.</p>	<p><a href="#">Contrast Scan Analyze</a></p> <p><a href="#">Contrast Scan Analyze README</a></p>

I want to use GitHub to...	Procedure	Related links
Ingest Contrast findings into GitHub Advanced Security	<ol style="list-style-type: none"> <li>Test the application whose information you want to include in a SARIF file with either of these methods: <ul style="list-style-type: none"> <li>Deploy the agent on the server deployed in the pipeline itself, where the test suit that should be part of the deployment runs. This action exercises the application and provides information for the SARIF file.</li> <li>Deploy the agent and exercise the application on a server before a test run.</li> </ul> </li> <li>Use the <a href="#">sarif</a> API to download the SARIF file from Contrast. You can filter the information by application and other metadata to customize the results.</li> <li>To ingest the SARIF file into GitHub Advanced Security, add the following code to your GitHub action file: <pre> - name: Upload SARIF file   uses: github/codeql-action/upload-sarif@v2   with:     sarif_file: results.sarif </pre> </li> </ol> <p>The Security &gt; Code Scanning tab shows the results.</p>	<a href="#">Uploading a SARIF file to GitHub</a>

## Integrate Contrast static SCA with GitHub

I want to use GitHub to...	Procedure	Related links
Run a static SCA scan against my repo using an action	Use the Contrast Security SCA GitHub action to scan your code repository for library vulnerabilities. You can also use the GitHub action to fail a build by updating your workflow file.	<a href="#">Contrast Security SCA GitHub action</a>
Ingest Contrast SCA findings into GitHub Advanced Security	<ol style="list-style-type: none"> <li>Test the application whose information you want to include in a SARIF file with either of these methods: <ul style="list-style-type: none"> <li>Deploy the agent on the server deployed in the pipeline itself, where the test suit that should be part of the deployment runs. This action exercises the application and provides information for the SARIF file.</li> <li>Deploy the agent and exercise the application on a server before a test run.</li> </ul> </li> <li>Use the <a href="#">sarif</a> API to download the SARIF file from Contrast. You can filter the information by application and other metadata to customize the results.</li> <li>To ingest the SARIF file into GitHub Advanced Security, add the following code to your GitHub action file: <pre> - name: Upload SARIF file   uses: github/codeql-action/upload-sarif@v2   with:     sarif_file: results.sarif </pre> </li> </ol> <p>The Security &gt; Code Scanning tab shows the results.</p>	<a href="#">Uploading a SARIF file to GitHub</a>

## Integrate IAST technology (Contrast Assess) with GitHub

I want to use GitHub to...	Procedure	Related links
Export results in a SARIF file and ingest it into GitHub Advanced Security	<ol style="list-style-type: none"> <li>Test the application whose information you want to include in a SARIF file with either of these methods: <ul style="list-style-type: none"> <li>Deploy the agent on the server deployed in the pipeline itself, where the test suit that should be part of the deployment runs. This action exercises the application and provides information for the SARIF file.</li> <li>Deploy the agent and exercise the application on a server before a test run.</li> </ul> </li> <li>Use the <a href="#">sarif</a> API to download the SARIF file from Contrast. You can filter the information by application and other metadata to customize the results.</li> <li>To ingest the SARIF file into GitHub Advanced Security, add the following code to your GitHub action file: <pre> - name: Upload SARIF file   uses: github/codeql-action/upload-sarif@v2   with:     sarif_file: results.sarif </pre> </li> </ol> <p>The Security &gt; Code Scanning tab shows the results.</p>	<a href="#">Uploading a SARIF file to GitHub</a>

I want to use GitHub to...	Procedure	Related links
Send vulnerabilities from Contrast directly to GitHub issues	Contrast supports the ability to send vulnerabilities to a GitHub repository and report them as issues. <a href="#">Integrate with GitHub (page 1080)</a> explains how to do this.	

## Integration example: Assess and GitHub

The example in this topic uses the Java agent. You can adapt this procedure for other Contrast agents.

### Before you begin

- Find a GitHub workflow that executes tests.
- Find where mvn is executed with a command similar to the following:

```
run : mvn -B clean install -Dmaven.gitcommitid.skip=true
```

- Add Assess to a test with a command similar to the following:

```
-DargLine="-javaagent:/tmp/contrast.jar --add-opens java.base/sun.net.spi=ALL-UNNAMED"
```

- The easiest method to download and configure an agent is to use the Add New agent wizard in the Contrast web interface for the language you want to use.

### Steps

- Download the agent with a command similar to the following:

```
curl -L https://download.java.contrastsecurity.com/latest -o contrast.jar
```

The Add New agent wizards in the Contrast web interface display the latest commands for downloading Contrast agents.

- Configure the agent with this environment variable as a [GitHub action secret](#) in the GitHub env workflow:



#### NOTE

The value of the environment variable is security sensitive, therefore configuring it as a secret is recommended.

```
env:
 # Setting an environment variable with the value of a configuration
 variable update test
 CONTRAST__ASSESS__ENABLED: true
 CONTRAST__PROTECT__ENABLED: false
 CONTRAST__API__TOKEN: <token-value>
```

You can retrieve the token value from the Contrast web interface under **Organization settings > Agent keys** or by using the [Contrast agent configuration editor \(page 109\)](#).

If you are using an older agent, configure these environment variables as GitHub action secrets:

```
env:
 # Setting an environment variable with the value of a configuration
 variable update test
 CONTRAST__API__URL: https://<yourURL>/Contrast
 CONTRAST__ASSESS__ENABLED: true
 CONTRAST__PROTECT__ENABLED: false
```

```
CONTRAST__API__API_KEY: ${ secrets.CONTRAST__API__API_KEY }}
CONTRAST__API__SERVICE_KEY: ${ secrets.CONTRAST__API__SERVICE_KEY }}
CONTRAST__API__USER_NAME: ${ secrets.CONTRAST__API__USER_NAME }}
```

## GitHub Actions

Contrast provides several GitHub actions that simplify a variety of analysis tasks. For example, you might find some of these GitHub Actions useful:

GitHub Action	Description
Contrast Verify	This GitHub Action verifies an application that uses Contrast by determining whether the application violates a Job Outcome Policy or threshold of open vulnerabilities
Contrast Security SCA	This GitHub Action lets you use Contrast to detect vulnerable dependencies in your code.
Contrast Security EKS Build Deploy	This GitHub Action builds and deploys a java application to the Amazon Elastic Kubernetes Service (EKS) with a Contrast Security Java Agent.
Contrast Scan Analyze	This GitHub Action lets you use Contrast Scan to find vulnerabilities in your code.
Contrast Security AKS Build Deploy	This GitHub Action builds and deploys a Java application to the Azure Kubernetes Service (AKS) with a Contrast Java agent.
Contrast Security Azure Spring Cloud Deploy	This GitHub Action deploys a Java application with a Contrast Java agent to the Azure Spring Cloud PaaS environment.

The GitHub Marketplace contains the most current [GitHub Actions for Contrast](#).

## Gradle plugin

The Contrast Gradle plugin is used to integrate the *Contrast.jar* with your build. It's capable of authenticating to Contrast, downloading the latest Java agent and verifying your build.



### NOTE

- **Gradle** is a build tool that utilizes `build.gradle` files to configure your applications. It's used to build, package, and test various types of applications.

## Clone a sample web application

The easiest way to set up a project is to clone our sample Gradle-based web application. This application has been migrated from Maven to Gradle and relies on MongoDB.

1. Install and set up the database path.

```
git clone https://github.com/Contrast-Security-OSS/Contrast-Sample-Gradle-Application
brew install mongodb
sudo mkdir -p /data/db
brew services start mongodb
```

2. An application is ready to run. Open the *Contrast-Sample-Gradle-Application/build.gradle* file. Scroll to find the `contrastConfiguration` extension. You can find all of the values in your [personal keys \(page 598\)](#) except `appName` and `serverName`.

```
contrastConfiguration {
 username = "username"
 apiKey = "apiKey"
 serviceKey = "serviceKey"
 apiUrl = "apiUrl"
```



```
orgUuid = "orgUuid"
appName = "editLATER"
serverName = "editLATER"
}
```

3. Install the Contrast JAR file by calling the `contrastInstall` task. This installs the Contrast JAR in the project's build directory.

```
cd path/to/Contrast-Sample-Gradle-Application
gradle build -x test contrastInstall
```

4. Run the application with the Java agent. The server starts.

```
cd path/to/Contrast-Sample-Gradle-Application/build
java -Dcontrast.agent.java.standalone_app_name=mytestapp
-Dcontrast.server=mytestserver -jar libs/Contrast-Sample-Gradle-
Application-0.0.1-SNAPSHOT.jar
```

5. Check that the application is running at `localhost:8080` and that the application shows up in Contrast.
6. In Contrast, verify that the application with the `appname` specified in the command above shows up.
7. In the Contrast-Sample-Gradle-Application project's `build.gradle`, edit the `contrastConfiguration` to specify the `appName` and `serverName` specified as options with the Java agent in the previous step.

```
contrastConfiguration {
 username = "alreadySetup"
 apiKey = "alreadySetup"
 serviceKey = "alreadySetup"
 apiUrl = "alreadySetup"
 orgUuid = "alreadySetup"
 appName = "mytestapp"
 serverName = "mytestserver"
}
```

8. Run the verification task at any time to check for vulnerabilities.

```
gradle build contrastVerify -x test
```

## Use the plugin

The plugin code can be viewed in our [GitHub repository](#). Here you can review the two tasks added by the plugin, `contrastInstall` and `contrastVerify`, and how they work.

The latest version of the plugin can be found on the [Gradle plugin webpage](#).

Task	Description
<code>contrastInstall</code>	<p>Installs a Contrast Java agent to your local project. The plugin edits the <code>org.gradle.jvmargs</code> property in the <code>gradle.properties</code> file to launch the JVM with the Contrast agent. An application version, by which the vulnerabilities are filtered in the <code>contrastVerify</code> task, is generated during this task. The plugin generates the application version in the following order:</p> <ul style="list-style-type: none"> <li>• If your build is running in TravisCI, Contrast uses <code>appName=\$TRAVIS_BUILD_NUMBER</code>.</li> <li>• If your build is running in CircleCI, Contrast uses <code>appName=\$CIRCLE_BUILD_NUM</code>.</li> <li>• If your build is running in neither TravisCI nor CircleCI, Contrast generates one in the format <code>appName-yyyyMMddHHmm</code>. (Where <code>yyyyMMddHHmm</code> is the time of the build generation.)</li> </ul>
<code>contrastVerify</code>	Checks for new vulnerabilities in your web application.

## Contrast Security IntelliJ plugin

Use the [Contrast IDE](#) (Integrated Development Environment) plugin to integrate security vulnerabilities associated with projects ([Scan \(page 33\)](#)) and applications ([Assess \(page 28\)](#)). The Contrast plugin

provides thorough information about vulnerabilities visible in projects and applications on a near real-time basis. The plugin offers filters based on severity, status, and discovery date to customize the vulnerability data to view.

The key features include a vulnerability report to view vulnerabilities associated with applications, provide a tree view on the list of the vulnerabilities related to the current file open in the IDE with visual indicators based on criticality, provide in-depth details about each vulnerability, automate the collection of vulnerabilities for applications and projects based on a schedule.

## Before you begin

- Make sure you have the supported system requirements:
  - CPU: Quad-core
  - RAM: 16 GB
  - Storage: SSD, 128 GB
  - Monitor: 1080p
- Make sure you have the supported software requirements:
  - Operating systems: Ubuntu 22.04.5 LTS or Windows 11
  - JRE: 17.0.12
  - JVM Heap Size: -Xms 2g -Xmx 4g
- Use IntelliJ Community Edition 2024.3 and 2024.2

## Install Java Development Kit (JDK)

### Linux

1. Install [JDK 17.0.13](#) from the Oracle website.
2. Download the appropriate **tar.gz** file for your system (for example, **jdk-17.0.13\_linux-x64\_bin.tar.gz**).
3. Open a terminal and go to the directory where you downloaded the **tar.gz** file.
4. Extract the contents using the command (example):

```
tar -xvzf jdk-17.0.12_linux-x64_bin.tar.gz
```

5. Move the extracted JDK to the **/usr/lib/jvm/** directory:

```
sudo mv jdk-17.0.12 /usr/lib/jvm/
```

6. Set the **JAVA\_HOME** environment variable and update the **PATH**.
7. Edit **.bashrc** or **.profile** file (depending on the shell):

```
nano ~/.bashrc
```

8. Add the following lines at the end of the file:

```
export JAVA_HOME=/usr/lib/jvm/jdk-17.0.12
export PATH=$JAVA_HOME/bin:$PATH
```

9. Apply changes by reloading **.bashrc**.

```
source ~/.bashrc
```

10. Run the following command to verify that Java is correctly installed:

```
java -version
```

11. Continue by configuring the heap size.

## Windows

1. Install **JDK 17.0.13** from the Oracle website.
2. Download the appropriate **tar.gz** file for your system (for example, **jdk-17.0.13\_windows-x64\_bin.tar.gz**)
3. Go to the directory where you downloaded the **tar.gz** file.
4. Set the environment variables:
  - Add **JAVA\_HOME**
  - Select **Win + S**
  - Type **Environment Variables**
  - Select **Edit the system environment variables**
  - In the system properties window, click **Environment Variables**
  - Under system variables, select **New**:

```
Variable name: JAVA_HOME
Variable value: Path to your JDK installation (e.g., C:\Program
Files\Java\jdk17).
```

5. Locate the **Path** variable under **System Variables**.
6. Select **New** and add:

```
%JAVA_HOME%\bin
```

7. Select **OK**.
8. Verify the configuration and open a new command prompt:

```
java -version
```

9. Continue by configuring the heap size.

## macOS

1. Install **JDK 17.0.13** from the Oracle website.
2. Download the appropriate **tar.gz** file for your system (for example, **jdk-17.0.13\_macos-x64\_bin.tar.gz**).
3. Go to the directory where you downloaded the **tar.gz** file.
4. Extract the contents using the command (example):

```
tar -xvzf jdk-17.0.13_macos-x64_bin.tar.gz
```

5. Move the extracted folder to the JDK directory:

```
sudo mv jdk-17.0.13 /Library/Java/JavaVirtualMachines/
```

6. Set the **JAVA\_HOME** environment variable. Determine the JDK Path - the extracted JDK is now located at: **/Library/Java/JavaVirtualMachines/jdk-17.0.13/Contents/Home**.
7. Open the shell configuration file:

```
nano ~/.zshrc
```

8. Add the following lines to set the **JAVA\_HOME** and update the **PATH**:

```
export JAVA_HOME=/Library/Java/JavaVirtualMachines/jdk-17.0.13/Contents/
Home
export PATH=$JAVA_HOME/bin:$PATH
```

9. Save the changes.
10. Select **Ctrl+O** to save in nano.
11. Select **Ctrl+X** to exit.
12. Apply the changes:

```
source ~/.zshrc
```

13. Verify the configuration and open a new command prompt:

```
java -version
```

14. Continue by configuring the heap size.

## Configure the Heap Size

1. Open **IntelliJ IDEA**.
2. Go to **Help > Edit Custom VM Options**.
3. Add or update the following lines:

```
-Xms1024m # Initial heap size (1 GB)
-Xmx4096m # Maximum heap size (4 GB)
```

4. Save the file and restart **IntelliJ IDEA**.

## Install the Contrast IntelliJ plugin

Select one of these options.

### Install the Contrast IntelliJ plugin via JetBrains Marketplace

1. Open **IntelliJ IDE** and go to **File > Settings > Plugins**.
2. Under marketplace search for **Contrast** then select the install button.
3. After installation select **Apply** and then **OK**.

### Install the Contrast IntelliJ plugin via manual installation

1. Download the plugin's `.zip` file and download it to your machine.
2. Launch **IntelliJ IDEA** on your system.
3. Go to **Plugin Settings**:
  - Go to **File > Settings**
  - In the *Settings/Preferences* dialog, select **Plugins**
4. Install the plugin from the disk:
  - Click the gear icon in the top-right corner of the plugins window
  - Select **Install Plugin from Disk** from the drop-down menu
5. In the window, go to the location of the `.zip` file and select it.  
IntelliJ IDEA will verify and install the plugin. If prompted, confirm the installation.
6. Restart IntelliJ IDEA:
  - After the installation, a prompt may appear asking you to restart IntelliJ IDEA
  - Click **Restart IDE** to complete the installation process
7. Verify Installation:
  - Once IntelliJ IDEA restarts, go back to **Plugins** under Settings/Preferences
  - Ensure the plugin is listed under **Installed Plugins** and is enabled

## Configure the Contrast IntelliJ plugin

1. In IntelliJ IDEA, find the Contrast plugin and select the **Settings** icon (⚙️).
2. Enter:
  - **Contrast URL**: The URL of your Contrast instance. For example, `https://app.contrastsecurity.com`.
  - **Username**: the username or e-mail address that you use for logging into Contrast.
  - **API Key, Service Key, and Organization ID**: Copy these values from your user profile in the Contrast web interface.

To find these values:

1. In the Contrast web interface, select your name in the upper right of the page.
2. Select **User settings**.
3. Copy the values for the API Key, Service Key, and Organization ID.



### IMPORTANT

Do not use the Agent keys (found under **user menu > Organization settings > Agent keys**). The access rights for the Agent keys are more limited than the API keys found under User Settings. The Agent keys won't work with the Contrast IntelliJ plugin.

3. Select **OK**.

## Contrast Jenkins integration

**Jenkins** is a continuous integration (CI) tool that automates the process of building, testing, deploying, and running applications.

With the Contrast plugin for Jenkins, you can add application security gates to this pipeline. These gates contain criteria that can fail the Jenkins job for a vulnerable application with a build result like "Failure" or "Unstable".



### TIP

You can view the plugin source code in the [Jenkins Github repository](#).

Use these versions to ensure compatibility:

Jenkins	Contrast-Jenkins plugin	Contrast
2.60.3	3.4	3.7.6
2.60.3	3.7	3.7.10
2.60.3	3.8	3.8.0

## Install and use Jenkins plugin

1. [Define a connection \(page 1090\)](#) between Contrast and Jenkins.
2. Depending on your situation, decide how you will use Jenkins:
  - If you are using freestyle jobs, you can define vulnerability security controls at a [system level \(page 1091\)](#) or as a [post-build action step \(page 1091\)](#).
  - Define vulnerability security controls for [pipeline steps \(page 1092\)](#).
  - Optionally, a Contrast Organization Administrator define a [job outcome policy \(page 1093\)](#).
3. [Run a build \(page 1097\)](#).

## See also

- [Connect with Jenkins \(page 1090\)](#)
- [Define security controls \(page 1093\)](#)
- [Define a job outcome policy \(page 1093\)](#)
- [Run a build \(page 1097\)](#)

## Connect

### Before you begin

- All Contrast Security integrations require an [organization admin role \(page 1267\)](#) in order to set up the connection.
- [Install Jenkins](#).
- Have already setup a [Jenkins pipeline](#).

### Define a connection

To define a connection to Contrast in Jenkins:

1. Select **Manage Jenkins** in the left sidebar of your Jenkins dashboard.
2. Select **Manage Plugins** under **System Configuration**.
3. Check to enable the **Contrast Continuous Application Security** plugin under the **Installed** tab.
4. Select **Manage Jenkins** again.
5. Select **Configure System** to find the **Contrast Connections** section.
6. Enter your **Contrast username**. Your username is the email address you use for your account in Contrast.
7. Enter the:
  - **Contrast API key**,
  - **Contrast service key**,
  - **Contrast URL** , and
  - **Organization ID**.You can find these in [your profile \(page 598\)](#) under **User settings > Profile > Your keys**.
8. Select a **Result of a vulnerable build** to choose how you want Contrast to mark your Jenkins job when your application is too vulnerable:
  - **Failure**
  - **Unstable**
  - **Success**
  - **Not\_built**
  - **Aborted**
9. Check the box next to **Apply this vulnerable build result to the job when Jenkins encounters an error with Contrast** if you want the Jenkins job to automatically fail whenever your Jenkins instance can't find your application.
10. You can define the criteria that the Contrast plugin will use to determine whether an application is too vulnerable at the Jenkins system level. Check the box next to **Allow global Contrast Vulnerability Threshold Conditions to be overridden in a Job configuration** if you want job level controls to override system level controls. Leave the box unchecked if you want to enforce consistency of criteria across all Jenkins jobs in your instance.



#### NOTE

If you are using a [job outcome policy \(page 1093\)](#) to set security controls, those policies will override any policies set at the job level or system level.

11. Click **Test Contrast connection** to make sure that the plugin can authenticate to Contrast and retrieve information about your applications' vulnerabilities.
  - A success message displays when plugin is authenticated.
  - If unsuccessful, check that the URL you received from Jira and the one you posted in Contrast are matching.

## See also

- [Define security controls \(page 1093\)](#)
- [Define a job outcome policy \(page 1093\)](#)
- [Run a build \(page 1097\)](#)

## Define security controls at a system level

After you [define a connection \(page 1090\)](#) in Jenkins, define if you are using freestyle jobs, you may want to set Contrast vulnerability security controls at a system level. Alternatively, you can [set security controls at a job level \(page 1091\)](#), or if you use a [job outcome policy \(page 1093\)](#) those security controls will take precedence.

1. Under **Contrast Connections > Contrast Vulnerability Security Controls**, select a **Connection** you have previously created, from the dropdown.
2. Set the **Number of Allowed Vulnerabilities**. This number is exclusive; if you set it to "5", Jenkins will fail if there are six or more vulnerabilities. This field is required.
3. Choose a **Vulnerability Severity** from the options in the dropdown. (These are the same [vulnerability severity options \(page 1039\)](#) in Contrast.) The plugin sets a filter in the API call for all vulnerabilities greater than or equal to this field. This field is not required, but selecting it will narrow your results. So if the number is set to "5" and the severity to **High**, Jenkins will fail if there are six or more critical vulnerabilities.
4. Choose a **Vulnerability Type** (rule name) from the dropdown. The plugin checks for the number of vulnerabilities with the rule type selected and compares it to the number of allowed vulnerabilities for that rule. This field is not required, but selecting it will narrow your results. You can choose one severity and one rule type per security control.
5. Choose from the list of **Vulnerability Statuses**. Statuses are not required but can be helpful. For example, select **Confirmed** and **Suspicious** only to return vulnerabilities with an open status. Leave this blank if you do not want to filter vulnerabilities by status.  
You can add multiple vulnerability security controls, but the plugin will fail the job on the first bad condition and set the build result on the first violated vulnerability security control.

## Define security controls as a post-build action step

After you have set security controls at the [system level \(page 1091\)](#) in Jenkins, you can also add security controls at a job level for freestyle jobs that are not part of a Jenkins Pipeline. To do this:

1. When defining a job in Jenkins, find the **Post-Build Actions** section.
2. Select a **Connection** you have previously created, from the dropdown.
3. Choose your application. This field is required.
  - If your application has been instrumented, select your application from the **Choose your application** dropdown.
  - If your application has not yet been instrumented, indicate your application using the **Application Name** and **Application Language** fields. You must provide the same application name in Jenkins that you will use when you do instrument your application. Contrast will use that same name and language during the post-build action step after the application has been instrumented.
4. If the connection is configured to [allow the system-level vulnerability security controls to be overridden \(page 1091\)](#), you can override that setting by checking the box next to **Override Vulnerability Security Controls at the Jenkins system level**.  
If you do this, you will also need to indicate the **Number of Allowed Vulnerabilities**, **Vulnerability Severity**, **Vulnerability Type**, and **Vulnerability Statuses** for this job.
5. Select how you want to query vulnerabilities by selecting an option under **Query vulnerabilities by**. That way, only those vulnerabilities found from that job will be considered. By default, the plugin uses the first option: `appVersionTag, format: applicationId-buildNumber`.

## Define vulnerability security controls for pipelines in Jenkins

You can use the `contrastAgent` pipeline step to download the Contrast agent, then instrument and exercise your application. You can use the `contrastVerification` pipeline step to verify your application and set parameters for a security control.

### Download with `contrastAgent`

A pipeline step with the name `contrastAgent` downloads the latest Contrast agent.

Parameter	Required	Description	Examples
<code>profile</code>	Required	Contrast connection profile used to communicate with Contrast	<code>MyConnection</code>
<code>outputDirectory</code>	Required	This defines where to put the downloaded agent.	<code>env.WORKSPACE</code>
<code>agentType</code>	Required if <code>applicationId</code> is not defined.	Type of agent used to instrument the application (not case sensitive)  Options are: Java, .NET, .NET_Core, Node, Ruby, Python	Java

Here is an example of how to add a pipeline step with the name `contrastAgent` :

```
node{
 stage('Download Latest Contrast Agent'){
 contrastAgent profile:'MyConnection', outputDirectory: env.WORKSPACE,
 agentType: 'Java'
 }
}
```

### Verify application with `contrastVerification`

You can use a pipeline step with the name `contrastVerification` to verify whether an application is vulnerable.

Parameter	Required	Description	Examples
<code>profile</code>	Required	Use <code>profile</code> to specify the connection used to communicate with Contrast.	Contrast Connection
<code>queryBy</code>	Required	Use <code>queryBy</code> to filter build-related vulnerabilities. For options 1, 2 and 4, this value must match the <code>contrast.override.appversion</code> parameter that was passed to the Contrast agent when running your application.  Enter the option number for how you want to query vulnerabilities (defaults to 1):  <ol style="list-style-type: none"><li>1. <code>appVersionTag</code>, format: <code>applicationId-\${BUILD_NUMBER}</code></li><li>2. <code>appVersionTag</code>, format: <code>applicationId-\${JOB_NAME}-\${BUILD_NUMBER}</code></li><li>3. <code>startDate</code> (This is the build timestamp. It only looks at vulnerabilities discovered after the build starts.)</li><li>4. <code>APPVERSIONTAG</code> (This is the job parameter or environment variable. Select this option if you want to specify your own text, then export <code>APPVERSIONTAG</code> as an environment property within your Jenkins job. Both <code>JOB_NAME</code> and <code>BUILD_NUMBER</code> are already available as Jenkins environment properties.)</li></ol>	1



Parameter	Required	Description	Examples
applicationId	Required, if applicationName and agentType are not defined.	The ID of the application or application module you are trying to verify	cb3ea678-38c8-4487-ba94-692a117e7966
applicationName	Required, if applicationId is not defined	The name of the application you are trying to verify (not case sensitive)	MyApp
count	Optional	The total number of allowed vulnerabilities, defaults to 0	10
rule	Optional	Defaults to <b>All</b>	xss
severity	Optional	Defaults to <b>All</b> . Other options are <b>Critical</b> , <b>High</b> , <b>Medium</b> and <b>Low</b> .	<b>High</b>
appVersionTag	Optional	The value that was passed to the <code>contrast.override.appversion</code> parameter of the Contrast agent	v1.2.3

Here are some examples of how to add a pipeline step with the name `contrastVerification`:

- **Use `queryBy` `startDate`:**

```
contrastVerification applicationId: '1e6ad9c6-89d4-4f06-bdf6-92c569ec89de', count: 1, profile: 'new-profile', queryBy: 3, rule: 'cache-controls-missing', severity: 'High'
```

- **Use `queryBy` custom `appVersionTag` parameter:**

```
contrastVerification applicationId: '1e6ad9c6-89d4-4f06-bdf6-92c569ec89de', count: 1, profile: 'new-profile', queryBy: 4, appVersionTag: 'v1.2.3' rule: 'cache-controls-missing', severity: 'High'
```

- **Use `applicationName` and `AgentType` to define the application:**

```
contrastVerification applicationName: 'MyApp', agentType: 'Java', count: 1, profile: 'new-profile', queryBy: 3, rule: 'cache-controls-missing', severity: 'High'
```

- **Verify an application with a preset or overridden vulnerability security control.**

If you know that the vulnerability security control has been preset in [Contrast \(page 1093\)](#), then you only need to define the profile and either the `applicationId` or (`applicationName` and `agentType`):

```
contrastVerification applicationId: '35ae7b89-1c76-414b-b317-c444ce27608b', profile: 'ContrastConnection'
```

## Jenkins security controls

You can define security controls for Jenkins:

- [At a system level \(page 1091\)](#)
- [As a post-build action step \(page 1091\)](#)
- [For pipelines \(page 1092\)](#)

## Define a job outcome policy

Job outcome policies (supported in the Contrast Jenkins integration version 3.3 and later) assign build outcomes to Jenkins jobs that use the Contrast plugin. These policies mark jobs with a build outcome status such as **Failure**, **Unstable**, or **Success** based on criteria you set.

## Before you begin

You must be an Organization Administrator to define a job outcome policy in Contrast.

## Define a policy

To define a job outcome policy:

1. Under [organization settings \(page 1158\)](#), select **Integrations** in the left navigation.
2. In the **Jenkins** row, select **Add job outcome policy**.

### Define Jenkins job outcome policy


Job outcome policies assign a build outcome to Jenkins jobs for the applications you choose and vulnerability properties you define. [Learn more](#)

Name \*

Applications

**Vulnerability properties**

Environments

Vulnerability rules and severities 

[+ Add another rule](#)

Vulnerability statuses

Vulnerability first seen

From

Until

☐ Apply the **Query vulnerabilities** by selection from the plugin when filtering vulnerabilities.


**Policy outcome**

If the policy is violated, assign a job outcome of:

☐ Failure

☒ Unstable

☐ Success

 For applications with conflicting job outcome policies, the policy with the most severe outcome will apply.

☒ Enable this job outcome policy

3. Define a name for the job outcome policy (required).
4. Under **Applications**, indicate the applications to which the policy should apply. You can identify applications by their name, importance level, or tag. If you select by application name, you can select individual or [merged applications \(page 613\)](#).
5. Under **Vulnerability properties**, define a limit of which vulnerabilities, in which environments will be included in the policy. Use the **Environment**, **Vulnerability status** and **Vulnerability first seen** to filter the vulnerabilities that you want to include in this policy. Use the **Vulnerability rules and severities** to set a threshold for how many of those rules (at a particular severity) will trigger the outcome status change.
  - **Environment(s)**: Select the environment where you want to apply the policy. For example, to block vulnerabilities from moving from test (QA) to production, select **QA**. (However, if you do that, vulnerabilities in the development environment are not considered. Select **Development** to also include those vulnerabilities. Or select **All environments** if you want vulnerabilities from all environments to be included.)

- **Vulnerability statuses:** Select which statuses will be included. [Statuses \(page 1035\)](#) are determined by Contrast.

**TIP**

In most cases, you should only select open statuses like **Reported**, **Confirmed**, and **Suspicious** (rather than closed statuses like **Not a problem**, **Remediated** or **Fixed**). That way, Jenkins jobs won't fail or turn unstable due to vulnerabilities that developers have already remediated.

- **Vulnerability first seen:** Set a time range for the vulnerabilities you'd like to include in this policy according to when they were first seen.

Use **From** and **Until** fields to set the beginning and end of the time range. Select the beginning of the time range to be: the **Job start time**, a predetermined period of time before the Contrast step runs, or the day of **Application onboarding**. Select the end of the time range to be a predetermined number of days before the Contrast step runs, or until the Contrast step runs in Jenkins, or choose an option for a specific amount of time. You can also select **Custom** to choose a specific date for either field.

If vulnerabilities were first found outside of this time range, they will not be included in the policy.

**TIP**

To incentivize developers to remediate vulnerabilities within a time period (for example, a week), define the policy so that only vulnerabilities found more than 7 days ago would be considered for policy violation. To do this, set **From** to **Application onboarding** and **Until** to **Last 7 days**.

- **Vulnerability rules and severities:** Use this section to set a threshold for the number, type and severity of vulnerabilities you want the policy to allow.

For each rule, select the **Severity** or **Assess rule type**, and then the **Number of allowed vulnerabilities**. Select **Add another rule** to add multiple rules.

The **Number of allowed vulnerabilities** determines how many vulnerabilities of this severity will be permitted without affecting this build. If you set it to "0", then a single vulnerability will change the build outcome status. If you set it to "10", then the build outcome status won't change until 11 vulnerabilities of that type are found. If you leave the **Number of allowed vulnerabilities** blank for a specific rule type or severity, it will allow *all vulnerabilities* of that rule type or severity.

For example, if you set this to **All rules** and 1 vulnerability, any single vulnerability would trigger the policy. You could also limit this policy to 5 critical vulnerability rules and 2 cross-site-scripting vulnerability by adding another rule.

- Check the box next to **Apply the "Query vulnerabilities by" selection from the plugin when filtering vulnerabilities**. You can define how to query vulnerabilities in a Jenkins job either using the Contrast Assess [post build step \(page 1091\)](#) or [pipeline step \(page 1092\)](#). For example, you can use the `AppVersionTag`, or the date when the vulnerability was last seen. If this checkbox is checked, then the query is included when the job outcome policy is evaluated.

This example shows possible rules and settings for a job outcome policy that will change the outcome status in Jenkins if these conditions are met.

## Vulnerability properties

### Environments

QA ×

### Vulnerability rules ?

All rules ▾

Critical ▾

SQL Injection ▾

### Number of allowed vulnerabilities ?

3 vulnerabilities

0 vulnerabilities ×

- vulnerabilities ×

+ Add another rule

### Vulnerability statuses

Reported × Confirmed × Suspicious ×

### Vulnerability first seen

From

Application onboarding ▾

Until

The Contrast step runs i... ▾

With this example, the following vulnerabilities will be considered for policy violation:

- All vulnerabilities found on a server designated as a QA environment.
- All vulnerabilities that have a status of **Reported**, **Confirmed** or **Suspicious**.
- Any vulnerability that was first discovered between application onboarding and when the Contrast step runs in Jenkins.

The policy will be violated and the outcome status will change, if at least ONE of these occur:

- There is at least 1 **Critical** vulnerability.
- There are at least a combined total of 4 **High**, **Medium**, **Low** or **Note** severity vulnerabilities of any rule type except SQL injection.



### NOTE

Vulnerabilities are only counted once, with precedence given to the most specific setting (for example, a particular rule type) to the least specific (**All rules**). If vulnerability limits are set for both a rule type and its severity level, the vulnerabilities will be included in the rule type count, but not in the severity's vulnerability count. So in this example, **Critical** vulnerabilities are counted under severity, but **High**, **Medium**, **Low** and **Note** severities are combined under **All rules**.

6. Under **Policy outcome**, select the outcome of the policy. Contrast marks jobs that match the selected criteria as **Failure**, **Unstable**, or **Success**. For applications with multiple job outcome policies, the most severe outcome from all violating policies will apply.
7. At any point, you can use uncheck the box next to **Enable this job outcome policy** to pause Contrast from enforcing policies on Jenkins jobs without having to delete the individual policy.

## Run a build

To run a build for the first time:

1. In Jenkins, select the job or project you want to run.
2. In the menu on the left, select **Build Now**.
3. To see more details, you can view the log output.
4. If you are using a freestyle job, you can view data from the task. Select the run and, in the left menu, select **Vulnerability Report**.



### IMPORTANT

You may also see a chart in the job or project overview, however the chart is not visible if you used a Contrast pipeline step, or if at least one of the applications selected is being overridden by a job outcome policy.

## Integrate with Jira

Integrate Jira with Contrast to automatically generate tickets, synchronize comments and push notifications for your applications.

Before you begin, you must have:

- Jira Cloud or Jira 8.
- Jira account credentials. For Jira Cloud, this is username and API key. For on-premises Jira installations, this is username and password.
- Permission to create issues in the target project.
- A running Jira instance accessible via HTTP to Contrast.
- A project to associate with an application that is already instrumented in Contrast.

### See also

- [Connect with Jira \(page 1097\)](#)
- [Configure Jira for Assess \(page 1098\)](#)
- [Configure Jira for Serverless \(page 1100\)](#)
- [Manage Jira credentials \(page 1101\)](#)

## Connect to Jira

To integrate Jira with Contrast:

### Set up Contrast for Jira

1. In Contrast, go to the **user menu > Organization settings > Integrations**.
2. Select **Connect** for the Jira integration.
3. In the **Setup Contrast with Jira** section, add a name for the Jira integration, the username and the API key (or password for Jira that is on-premises only). Add the URL for the Jira instance, and be sure that Contrast can access the URL.
  - Enter a **name** for the Jira integration.
  - Add the **URL** for the Jira instance.
  - Select **Assess** or **Serverless**.



### NOTE

Contrast saves the username, API key or password, and the URL for Jira as a set of credentials for this integration.

4. Select **Test connection**. The test may take a few moments, if you have many Jira projects. The test confirms that Contrast can reach the specific Jira instance and the user can log in.
5. After testing the connection, configure [Jira for Assess \(page 1098\)](#) or [Jira for Serverless \(page 1100\)](#).

## See also

- [Configure Jira for Assess \(page 1098\)](#)
- [Configure Jira for Serverless \(page 1100\)](#)
- [Manage Jira credentials \(page 1101\)](#)

## Configure Jira for Assess

After testing your Jira connection, you can configure Jira to create tickets based on triggers you've set.

## Before you begin

- Successfully [connected with Jira \(page 1097\)](#)

## Steps

1. After Contrast connects to Jira, select **Applications** to add the Contrast applications that will trigger Jira tickets for security issues. You can also trigger Jira tickets only for applications with specific importance levels in Contrast. Select **Application importance** and add the application levels you want to use as a filter for Jira tickets.

The screenshot shows the 'Jira Connection' configuration dialog within the 'Organization Settings' section. The dialog is titled 'JiraConnection' and has a close button (X) in the top right corner. It contains the following fields and options:

- Connection Name\***: A text input field with the value 'howDoesThiswork'.
- Credentials Name**: A dropdown menu showing 'https://mycompany.atlassian.net-user1' with a 'Manage credentials' link next to it.
- Integration Type**: Three radio buttons: 'Assess' (selected), 'Serverless', and 'Applications'.
- Applications**: A dropdown menu showing 'All Applications (95)'.
- Application Importance**: A radio button option.
- Project Name**: A dropdown menu showing 'FAKEBUG'.
- Default Epic**: A dropdown menu showing 'No default epic'.
- Default Assignee**: A dropdown menu showing 'Unassigned'.
- Default Issue Type**: A dropdown menu showing 'Task'.
- Default Priority based on vulnerability severity**: A table with five rows:
 

Severity	Priority
Critical	Critical
High	Critical
Medium	Major
Low	Standard
Note	Standard
- Additional Jira Fields**: A dropdown menu showing 'Labels' and a '+ Add Jira field' link.
- Default Value**: A text input field with the value 'sensitive-data-masking' and a clear button (X).
- Checkboxes**:
  - ☐ Allow exclusion of sensitive information from issue titles and additional fields when creating tickets
  - ☐ Enable two-way integration
  - ☐ Automatically create tickets for new vulnerabilities discovered
  - ☐ Allow sending tags as labels when creating tickets.
- Buttons**: 'Delete Configuration' (trash icon), 'Cancel', 'Connected' (checkmark icon), and 'Save'.

2. Use the **Project name**, **Default epic**, **Default assignee** and **Default issue type** fields to set custom values for Jira tickets that Contrast creates. You can also map vulnerability severity levels in Contrast to Jira priority values to help teams refine security tickets. If you want to prefill additional Jira fields, select **Add Jira field**. Use the dropdowns to select the fields you want to add and the default value for the field.

**NOTE**

Changing the **Project name** or **Default issue type** also changes the related Jira fields and values available to you. Contrast will keep any selected values that also apply to the new project or issue type.

3. Select the **Allow exclusion of sensitive information from issue titles and additional fields when creating tickets** option to enable the ability to mask the sensitive information being sent to Jira.
4. Select the option to **Enable two-way integration**:
  - If you want to change the vulnerability status in Contrast, whenever an issue closes or reopens in Jira. In Contrast, use the Vulnerability status dropdowns to configure how a Jira ticket status update will change the vulnerability resolution status.

**NOTE**

If you choose **Not a problem** as a status, Contrast requires you to enter a **Reason** in the dropdown. The default selection in the dropdown is **Other**.

- If you want to synchronize comments added to the Vulnerability **Activity tab** ([page 1022](#)) to the Jira issue and, the other way around, if a comment is added to the Jira issue it appears in the **Activity tab** under the Vulnerability.

- If you want to show the session metadata associated with the vulnerability in the Jira ticket.

This generates a URL visible below the checkbox, which your Jira administrator must use to [register and configure a webhook in Jira](#). Use the *Comment...* actions to synchronize the comments between the **Activity tab** and the Jira issue.

After you save the two-way integration, Contrast automatically tracks any status changes, added comments, or session metadata on related Jira tickets. You will see these as comments in the **Activity tab** for the vulnerability. Each comment includes the name of the Jira integration and a link to the ticket.

**NOTE**

Atlassian has deprecated the ability to register webhooks with non-https URLs. Therefore, Contrast on-premise users need to [configure HTTPS](#) ([page 1203](#)) before attempting to enable Jira two-way integration.

5. If you want a new Jira ticket made when Contrast discovers a vulnerability, select the option to **Automatically create tickets for new vulnerabilities discovered**. Then select which **Severity levels** or **Rules** should trigger new Jira tickets.

If Contrast creates a single Jira ticket for multiple vulnerabilities, the ticket status applies to all vulnerabilities associated with the ticket. If Contrast creates multiple tickets for a single vulnerability, all Jira tickets must close before Contrast can close the vulnerability.

**NOTE**

Automation options are not retroactive and will not generate Jira tickets for past vulnerabilities.

6. If you want to send tags as part of the ticket information that Contrast sends to Jira, select the option to **Allow sending tags as labels when creating tickets**.  
When you add tags to vulnerabilities and turn on this setting, Contrast sends tags to Jira as labels.

7. Select **Save** and begin using your Jira integration. To remove the integration select **Delete configuration**.

## Configure Jira for Serverless

After testing your Jira connection, you can configure Jira to choose a subset of severities and result categories from which the integration should create a Jira ticket.

### Before you begin

- Make sure you are [connected with Jira \(page 1097\)](#) and have AWS accounts in Contrast

### Steps

1. After Contrast connects to Jira, select **Add configuration** to configure the AWS accounts from which results will have Jira tickets created.

The screenshot shows the Contrast web interface. The top navigation bar includes the Contrast logo, tabs for Applications, Scans, Servers, Libraries, and Vulnerabilities, a search bar, and user profile icons. The left sidebar lists various settings categories, with 'Integrations' selected. The main content area displays the 'Organization Settings' page, specifically the 'Jira Connection' configuration form. The form includes fields for 'Connection Name \*' (with a placeholder 'Name this Jira integration'), 'Credentials Name \*' (with a placeholder 'robOwens' and a 'Manage credentials' link), radio buttons for 'Assess' and 'Serverless' (with 'Serverless' selected), an 'Accounts' section with a dropdown for 'All Accounts (0)', and four dropdown menus for 'Project Name', 'Default Epic', 'Default Assignee', and 'Default Issue Type'. Below these is a checkbox for 'Jira will automatically create tickets for new vulnerabilities discovered'. At the bottom, there is a 'Results' section with checkboxes for 'Critical' and 'High'. The form has 'Cancel', 'Connected' (with a green checkmark), and 'Save' buttons at the bottom right.

2. Enter the following details:
  - **Connection Name:** Choose a name for the Jira integration you are creating
  - **Credentials Name:** Enter the desired credentials to be used for authentication
3. Click the **Manage Credentials** link above the *Credentials Name* field to edit or create new credentials.
4. Enter the following details:
  - Credentials Name: the name you entered in the previous step
  - URL: Provide the URL for your Jira instance
  - Username
5. Enter the **API Key**. To authenticate calls from Contrast, you need to provide an API Key for Jira.
  - Click the information icon next to the API Key field name and click the *Get Started* link.
  - Click Create API Token.
  - Enter a name.
  - Click Create.
  - Click Copy to copy the API token to your clipboard.



- Go back to Contrast and paste the API token into the API Key field.
- 6. Click **Test Connection** to test the connection.
- 7. Click **Save** or **Done**.
- 8. Select the **Serverless** option to begin creating Jira tickets for vulnerabilities related to specific applications or accounts.
- 9. Enter the desired **Accounts** for which Jira tickets should be created.
- 10. Choose a Jira **Project** in which Contrast should create tickets.
- 11. Set the **Default Epic**. Specify the Epic under which the new tickets should be created (if applicable).
- 12. Set the **Default Assignee**. Specify the person who should be assigned to the tickets.
- 13. Set the **Default Issue Type**. Choose the default issue type for the created tickets (for example, Task / Story / Bug).

**NOTE**

Changing the **Project** or **Default issue type** also changes the related Jira fields and values available to you. Contrast will keep any selected values that also apply to the new project or issue type.

- 14. Choose the type of **Results** for which Contrast should create tickets.
  - Permissions
  - Exploits
  - Dependencies (CVEs)
  - Severity (Note, Low, Medium, High, Critical)
- 15. After configuring the Default Issue Type, you need to provide additional Jira fields.
  - Select **Add Jira field**.
  - Add the Reporter, T-Shirt Size, and Work Type fields
- 16. Map the Contrast severity levels to the corresponding Jira severity levels (for example, Critical, Major, Standard) as needed.
- 17. Select **Save** and begin using your Jira integration. To remove the integration select **Delete configuration**.

Watch the following video for a demonstration of the integration steps.

<https://player.vimeo.com/video/827314961?h=f049a72b04>

Watch the following video for detecting a new vulnerability.

<https://player.vimeo.com/video/827318792?h=78a2e50f7f>

## Manage Jira credentials

Contrast saves the most recent credentials for a Jira integration to help you set up new connections faster. The username, API key or password, and Jira URL values you enter in your first configuration are the default credentials for the next Jira integration. Contrast will pre-populate the next Jira configuration with the default credentials, but you can modify these values. You can also manage saved sets of Jira credentials to update all affected configurations.

To create or edit a single configuration with credentials that are different from your default set:

1. Go to the **user menu > Organization Settings > Integrations**.
2. Select **Show configurations** to see the list of existing Jira integrations. Select the one you want to update.
3. Select **Manage credentials** to see the Jira connection configuration details.

4. In the **URL** field, use the dropdown to choose a set of saved credentials, or manually update the **URL**, **username**, and **API key** or **password**.
5. Once you've updated the fields, select **Test connection** to be sure the changes work.
6. Select **Save**.

**NOTE**

If you're using new credentials, you must choose to override the existing set of credentials under the given name, or save the new values as a new credential set under a different name.

To edit multiple Jira configurations at the same time:

1. In Contrast, go to the **user menu > Organization settings > Integrations**.
2. Select **Manage credentials** in the Jira Integration.
3. In the **Manage Jira credentials** form, select a set of saved credentials in the dropdown.
4. Edit the username, API key or password, or Jira URL.
5. Select **Rename** if you want to use a different name for the edited credentials.

**NOTE**

Any updates to a set of credentials will affect all configurations using that set.

6. Select **Test connection** to be sure the integration works.
7. Select **Save**.

**NOTE**

Any updates to a set of credentials will change all configurations that use this set.

To delete credentials:

1. In Contrast, go to the **user menu > Organization settings > Integrations**.
2. Select the **Manage credentials** option and then select **Delete credential**.
3. Optionally, select **Show configurations** to see the list of existing Jira configurations.
4. Select one of two methods for deletion of the credentials and configuration:
  - Select the delete icon at the end of the row
  - Select the configuration name and then select the **Delete Configuration** option at the bottom of the form
5. Click **Delete** in the confirmation window.

**See also**

- [Jira integration \(page 1097\)](#)
- [Connect to Jira \(page 1097\)](#)
- [Configure Jira for Assess \(page 1098\)](#)
- [Configure Jira for Serverless \(page 1098\)](#)

**Two-way integration with Jira**

You can use a two-way integration with Jira if you want to change the vulnerability status in Contrast every time an issue closes or reopens in Jira. When you select the option to enable two-way integration,

this generates a URL that appears below the checkbox, which your Jira administrator must use to [register a webhook in Jira](#).

In Contrast, use the Vulnerability status dropdowns to configure how a Jira ticket status update will also change vulnerability resolution status.

**NOTE**

If you choose **Not a problem** as a status, Contrast requires you to enter a **Reason** in the dropdown. The default selection in the dropdown is **Other**.

After you save the two-way integration, Contrast automatically tracks any status changes on related Jira tickets. You will see these as comments in the **Activity** tab for the vulnerability. Each comment includes the name of the Jira integration and a link to the ticket.

**NOTE**

Atlassian has deprecated the ability to register webhooks with non-https URLs. Therefore, Contrast on-premise users need to [configure HTTPS \(page 1203\)](#) before attempting to enable Jira two-way integration.

## Contrast Maven plugin

[Maven](#) is a build tool that builds, packages, and tests your Java applications.

The Contrast Maven plugin can integrate Contrast Assess and Scan into your project's Maven build.

Use the [Contrast Maven Plugin Reference Documentation](#) for more details on:

- [Goals](#)
- [Usage](#)

### Goals

- **Scan:** The scan goal analyzes the Maven project's artifact with Contrast Scan to find vulnerabilities using static analysis.
- **Install:** The install goal includes the Contrast Java agent in integration testing to provide Contrast Assess runtime security analysis. For this goal to succeed, you need the [Organization Edit role \(page 1267\)](#).
- **Verify:** The verify goal verifies that none of the vulnerabilities found by Contrast Assess during integration testing violate the project's security policy (fails the build when violations are detected).

### See also

- [Contrast Scan \(page 632\)](#)
- [Install the Java agent \(page 123\)](#)

## Integrate with Microsoft Teams

Use the Microsoft Teams integration to receive notifications from Contrast in your configured Microsoft Teams instance.

To use this integration, configure a Power Automate instant cloud flow using Microsoft Teams.



### NOTE

As of January 31, 2025, Microsoft is changing the way you create webhook integrations with Microsoft Teams. This procedure describes the new method for integrating Contrast with Microsoft Teams.

### Before you begin

- A Microsoft Power Automate account is required.

### Create an instant cloud flow in Microsoft Teams

1. Sign in to Power Automate.
2. Select **Create > Instant cloud flow**.
3. Select this trigger: **Power Automate Integration with Microsoft Teams**.
4. Select **When a Teams webhook request is received**.  
This setting triggers your flow when a Teams channel receives a webhook request.
5. In Who Can Trigger the Flow, select **Anyone**.
6. Select **Create**.

### Configure the Instant cloud flow trigger

1. In the Actions tab, select **Add an Action**.
2. Select **Post card in a chat or channel**.  
This option sends information to a Teams channel in a visually appealing format.
3. In Team Selection, select a team where you want to post the card.
4. In Channel Selection, select the channel where you want to post the card.
5. In Adaptive Card, select **Attachments Adaptive Card**.
6. Save the flow.
7. in the Parameters tab for the flow, copy the HTTP URL.  
You need this URL to configure the Contrast integration.

### Configure the Contrast integration

1. In Contrast, under the **user menu**, go to **Organization settings > Integrations**.
2. In the Microsoft Teams row, select **Connect**.
3. Enter a name for the integration.
4. Paste the webhook URL copied from Microsoft Teams.
5. Select an application to enable notifications.
6. Select **Save**.

**NOTE**

Contrast disconnects this integration if it fails to return a successful response after five attempts.

## Integrate with PagerDuty

Integrate with PagerDuty incident management to receive attack notifications from Contrast.

To connect PagerDuty:

1. In the **user menu**, go to **Organization Settings > Integrations**.
2. Select **Connect** in the PagerDuty row.
3. In the window, enter a **Name** for the integration. This name will be displayed in notifications from Contrast.
4. In the **Message Severity** dropdown, choose the behavior of the alert. The default selection is "Critical." For more information about message severity, see the [PagerDuty documentation](#).
5. Enter an **Integration key**. To find your integration key to enter in this field, follow the steps in the [PagerDuty documentation](#).
6. In **Applications**, select the applications in your portfolio that you want Contrast to automatically generate incidents for within PagerDuty. The default selection is "All Applications."
7. Once you complete all of the fields, select **Test connection**. This process may take a few minutes, depending on the number of your PagerDuty projects. The test verifies that Contrast can reach the PagerDuty instance and that a message can be sent.

Manage notifications from your PagerDuty integration in [organization notifications \(page 1175\)](#).

**IMPORTANT**

Contrast will disconnect this integration if it fails to return a successful response after 5 attempts.

## Integrate with Solutions Business Manager

Integrate with Solutions Business Manager to receive notifications from Contrast.

Before you start, you must have:

- Solutions Business Manager account credentials (username and password)
- A running Solutions Business Manager instance accessible via HTTP to Contrast
- A project to associate the application instrumented by Contrast

To connect Solutions Business Manager:

1. Go to **Organization settings > Integrations** in the **user menu**.
2. Click **Connect** in the Solutions Business Manager row.
3. In the **Connect with Serena** page, enter a name for the bugtracker entry. It will be displayed when sending findings to bugtrackers.
4. Enter a username for the account connected to the Solutions Business Manager instance.

5. Enter the password for the username specified.
6. In the **Host** field, enter the URL to the Solutions Business Manager instance.
7. Select the application you would like to map to the Solutions Business Manager instance.
8. Enter the **Solutions Business Manager Project ID** to associate with this application.
9. Select **Test** to verify communication. This ensures that Contrast can communicate and authenticate with the instance, as well as verify the existence of the specified project.

## Integrate with ServiceNow

Integrate ServiceNow with Contrast to automatically generate incidents in ServiceNow.

Before you start, you must have:

- ServiceNow URL
- ServiceNow Username
- ServiceNow Password

## Connect to ServiceNow

To integrate ServiceNow with Contrast:

1. In Contrast, go to the **user menu** > **Organization settings** > **Integrations** page.
2. Select **Manage Credentials** for the ServiceNow integration.
3. Add the URL, username, and password for ServiceNow.

The screenshot shows the 'Integrations' page with the 'Service Now' integration selected. The form includes fields for 'URL' (http://service-now-url.com), 'Username' (username), and 'Password' (masked with dots). There are 'Manage Credentials' and 'Configure Applications' buttons at the top right, and 'Save' and 'Cancel' buttons at the bottom right.

4. Select **Save**.
5. Select **Configure Applications**.
6. Either enable the integration for all Assess applications or select specific application names from the dropdown list.

This screenshot shows the 'Configure Applications' step. It includes a toggle for 'Enable for all Assess Applications' (currently disabled) and a dropdown menu for 'Assess Applications' with three selected items: 'AppName1', 'AppName2', and 'AppName3'. 'Save' and 'Cancel' buttons are at the bottom right.

7. Select **Save**.

## Retry mechanism

If an event fails, it will be stored and retried every night at midnight GMT, increasing the retry count by one to a maximum of three for up to 72 hours. After the third unsuccessful retry, the event will be discarded.

If a `create vulnerability` event fails and is stored, any update or delete action relating to the failed event is stored and replayed in chronological order to help maintain the correct state.

## Integrate with Slack

With the Slack integration, you can receive [notifications \(page 1175\)](#) from Contrast in your configured Slack instance using a format similar to in-app notifications.

To connect Slack:

1. In Slack, go to your team's **Build** settings.
2. Add a new **Incoming webhooks** custom integration.
3. Choose the appropriate channel to which to send messages.
4. Copy the **Webhook URL**.
5. In Contrast, in the **user menu**, go to **Organization settings > Integrations**.
6. Select **Connect** in the Slack row.
7. Name the integration and paste the URL.
8. Selection the application for which you want to enable notifications.
9. Select **Save**.

To test the integration:

1. Go to **Organization settings > Notifications**.
2. In the dropdown under **Integrations**, select the Slack integration name.
3. For each **Subscription** (event type) you want to be notified of, click the toggle in the **Integrations** column.
4. Cause an event type to occur, and confirm that you get a notification in the Slack channel you specified.



### IMPORTANT

Contrast will disconnect this integration if it fails to return a successful response after 5 attempts.

## Integrate Splunk with Contrast

The Contrast Security App for Splunk provides actionable and timely application threat intelligence across your entire application portfolio. Contrast Security instrumented applications self-report these details about an attack:

- The attacker's IP address
- Authenticated username
- Method of attack
- Affected applications and servers
- Attack frequency and volume
- Level of compromise

### Before you begin

Ensure you have the following software:

- Splunk Enterprise or Splunk CloudPlatform, versions: 9.3, 9.2, 9.1, 9.0, 8.2, 8.1, 8.0
- Contrast Security App for Splunk 2.0.1 or later

- Applications instrumented with a Contrast agent

## Step1: Install Contrast Security App

For more guidance installing Splunk applications, visit their [documentation](#).

- **Install the app**

1. In Splunk, select **Apps**.
2. Search for Contrast Security App for Splunk.
3. Select **Install**.  
After installation, you should see the Contrast Security App in the Apps dropdown.

- **Install the app from a file**

1. Log in to [Splunkbase](#).
2. Search for the Contrast Security App.
3. Select download and save the file to a convenient location.
4. In Splunk, select **Apps**.
5. Select **Manage**.
6. Select **Install app from file**.
7. Select the app you downloaded.  
Optionally, select **Upgrade app** if you installed the Contrast Security app previously.
8. Select **Upload**.

## Step 2: Set up a syslog receiver in Splunk

Contrast agents stream SIEM events as UDP syslog events in CEF format.

1. In Splunk, select **Settings > Data Input**.
2. Add a new UDP listener.
3. Reuse port 514 or chose a different port.
4. For the source type, specify `contrast_events`.
5. Select **Save**.

UDP  
Data inputs > UDP

New

Showing 1-1 of 1 item

Results per page 25

UDP port	Source type	Status	Actions
8514	contrast_events	Enabled   Disable	Clone   Delete

## Step 3: Set up Contrast agents

- **Setup for a single server in the Contrast web interface**

1. In the header, select **Servers**.
2. Select a server that has Protect turned on.
3. Select the **Settings** icon (⚙️) in the top right of the page.
4. In Server settings, select **Enable output of Protect events to syslog**.
5. Specify the settings for the syslog output:



- Fully Qualified Domain name for the Splunk server where you want to send messages,
  - The UDP port
  - Syslog facility (the process that created the message)
  - The syslog message severity levels
6. Select **Save**.
- **Setup for all new servers in the Contrast web interface**
    1. From the user menu, select **Organization settings**.
    2. Select **Servers**.
    3. Select **Enable output of Protect events to syslog**.
    4. Specify the settings for the syslog output:
      - Fully Qualified Domain name for the Splunk server where you want to send messages,
      - The UDP port
      - Syslog facility (the process that created the message)
      - The syslog message severity levels
    5. Select **Save**.
  - **Setup with agent configuration files**

Specify the syslog settings in the agent configuration YAML file, as shown in this example:

```
agent:
 syslog:
 enable: true
 ip: splunk.mycompany.org
 port: 8514
 facility: 12
 # Set the log level of Exploited attacks. Value options are `ALERT`,
 # `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
 severity_exploited: CRITICAL

 # Set the log level of Blocked attacks. Value options are `ALERT`,
 # `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
 severity_blocked: WARNING

 # Set the log level of Probed attacks. Value options are `ALERT`,
 # `CRITICAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`.
 severity_probed: NOTICE
```

## Step 4: View Contrast data in Splunk

Splunk provides three dashboards where you can see Contrast data.

- In Splunk, under Apps, find **Contrast Security for Splunk**.
- Select **Dashboards**.
- Select a dashboard:
  - **Attack Dashboard:** This dashboard shows a summary of attacks that Contrast blocked in a specified time frame, identifies which applications are targeted for attacks, the type of attacks and when they occurred., and the most frequently targeted URIs.
  - **Attacks by Applications:** For the specified applications, this dashboard shows the number of attacks , the different types of attacks detected and blocked, the distribution of different attack types in a specified time frame, and the top 10 most attacked URIs.
  - **Attacks Geographical Distribution:** This dashboard shows a geographical view of attacks that Contrast Security detected and blocked.

## See also

These topics provide details for agent-specific syslog settings in agent configuration files:

[Configure Java \(page 149\)](#)

[Configure .NET Core \(page 290\)](#)

[Configure .NET Framework \(page 228\)](#)

[Configure Node.js \(page 352\)](#)

[Configure Python \(page 413\)](#)

[Configure PHP \(page 397\)](#)

[Configure Go \(page 534\)](#)

[Configure Ruby \(page 473\)](#)

## Integrate Contrast Security ADR with Splunk

The Contrast Security ADR integration with Splunk enables [ADR \(page 577\)](#) to send incident details to your SIEM (Security Information and Event Management), SOAR (Security orchestration, automation and response), and XDR (Extended Detection and Response) environments, contextualizing incidents with other threat detection and response solutions.

### How it works

When configured, the **Contrast Security ADR for Splunk** app sends detected attack events from the Contrast Security platform to a Splunk HTTP Event Collector.

The [Contrast Security ADR for Splunk](#) app on [splunkbase.com](https://splunkbase.com) enables Splunk to:

- Parse and normalize the data received over the HTTP Event Collector
- Display Contrast Security ADR dashboards, reports, and searches in Splunk
- (On request) Call the Contrast Security ADR REST APIs for contextual data to help investigate incidents
- Provide runbooks to assist SOC Analysts in resolving AppSec-related security incidents

### Before you begin

Before you start, you must have:

- Splunk Enterprise 9.2. See the [installation guide](#) for information.
- Splunk CIM (Common Information Model) 5.x and later
- Applications instrumented with a Contrast agent

## Step1: Install the Contrast ADR app in Splunk

### Install from the Marketplace

1. In Splunk Enterprise, select **Apps** and select **Find more apps**.
2. Search for **Contrast Security ADR for Splunk 1.0**.
3. Check the requirements.
4. Select **Install**.

After installation, you should see the **Contrast Security ADR for Splunk** app in the apps dropdown.

### Install from a file

1. In Splunk Enterprise, select **Apps** and select **Find more apps**.
2. Search for **Contrast Security ADR for Splunk 1.0**.

3. Select **download** and save the file to a convenient location.
4. Select **Manage**.
5. Select **Install app from file**.
6. Select the app you downloaded.  
Optionally, select **Upgrade app** if you previously installed the Contrast Security ADR app.
7. Select **Upload**.

After installation, you should see the **Contrast Security ADR for Splunk** app in the apps dropdown.

## Step 2: Set up Splunk CIM

1. In Splunk Enterprise, select **Apps** and select **Find more apps**.
2. Search for **Splunk Common Information Model (CIM)**.
3. Select **Install**.
4. Enter your Splunk.com mail and password credentials.
5. Select the **Accept** and **Login** buttons.

After installation, you should see the app in the apps dropdown.

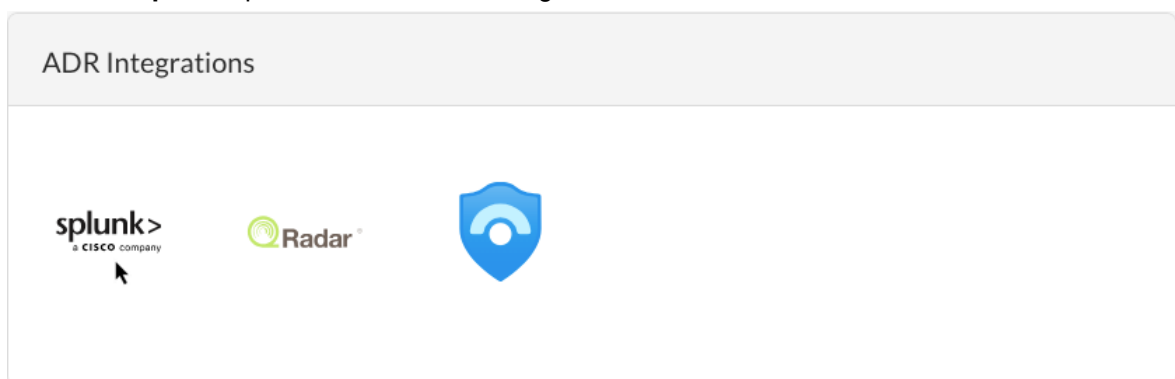
## Step 3: Set up an HTTP Event Collector input in Splunk

1. In **Splunk Enterprise**, go to **Settings > Data Input > HTTP Event Collector**.
2. Enter a **Name** to help identify the token receiving data over HTTP. Remember this name.
3. Select the **New Token** button at the top of the page.
4. Enter the fields and select **Next**.
5. Choose **Select source type** and specify `contrast:adr` for the source type.
6. Select the preferred index to store the data, such as `contrast`. If an index does not exist, create an index following [these steps](#).
7. Select the **Review** button.
8. Select the **Submit** button.
9. Copy the token value on the success page. This will be needed for the integration.

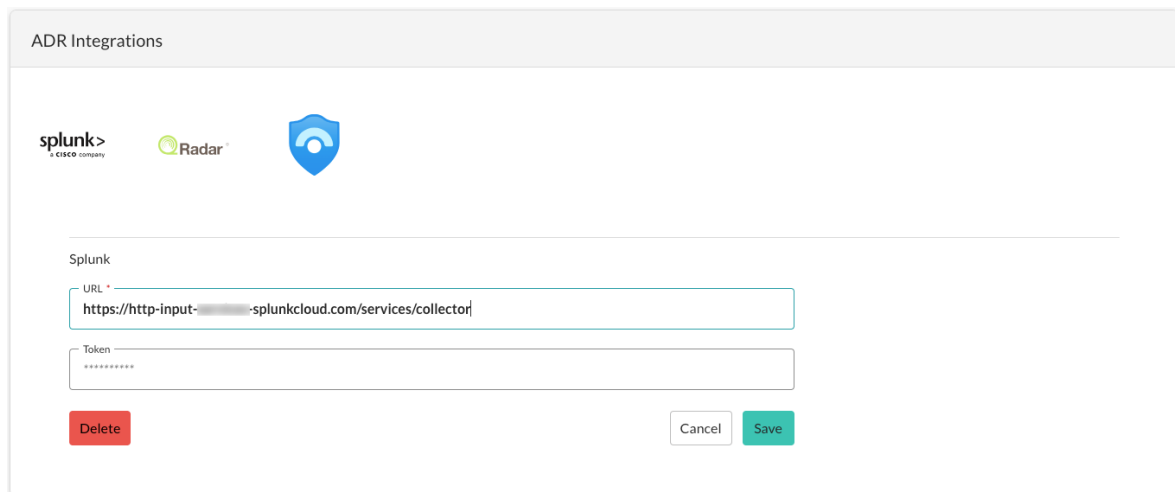
## Step 4: Configure Contrast Security ADR to send Attack Events to Splunk

Configure the integration in Contrast to send attack events to the Splunk app.

1. In Contrast, go to the **user menu** and select **Organization settings > Integrations**.
2. Select the **Splunk** option under the ADR Integrations section.



3. Under the Splunk fields, enter the URL and token information for the destination HTTP Event Collector as configured in [Step 3 \(page 1111\)](#). Note that the URL in the screen example is shown only for illustration purposes; refer to the [official Splunk Documentation](#) for more information.



ADR Integrations

splunk> Radar

Splunk

URL

Token

Delete Cancel Save

## Step 5: Set up macros to search for events

Macros are used to keep track of the index where Contrast Security ADR events are stored. The macro will be used for CIM mapping and correlation searches.

1. In **Splunk Enterprise**, go to **Settings > Advanced Search**.
2. Select **Search macros**.
3. Select **Add new**.
4. Search for the `contrast_search` macro.
5. Select the macro name to edit it.
6. Call the index name as the index provided in the HEC input as configured in [Step 3 \(page 1111\)](#).
7. Select **Save** to update the macro in the index where Contrast Webhook data is stored.

To validate the macro:

1. In **Splunk Enterprise**, select **Contrast Security ADR for Splunk 1.0** from the apps menu.
2. Select **Search**.
3. In the search box, type `contrast_search`.
4. Set the time range input to **All Time**.
5. Select **Search**.

You should be able to see the Contrast ADR events once the search is complete.

## Step 6: Set up CIM data models

Set up an Intrusion Detection Macro to recognize the Contrast ADR events.

1. In **Splunk Enterprise**, go to **Settings > Advanced Search > Macros**.
2. Select the **Splunk Common Information Model (CIM)** app.
3. Search for `cim_Intrusion_Detection_indexes` and select to edit.
4. Edit the definition as `eventtype=contrast_adr`.
5. Select **Save**.

## Step 7: Set up API details

1. In **Splunk Enterprise**, select **Contrast Security ADR for Splunk 1.0** from the apps menu.
2. Go to the **Setup > Setup Configurations** page.
3. Specify the settings in each field:
  - **Hostname:** The host domain of your Contrast platform. For example, `https://cs001.contrastsecurity.com`

- **Username:** The [username \(page 598\)](#) in Contrast
- **Organization UUID:** The [organization ID \(page 598\)](#) in Contrast
- **API Key:** The [API key \(page 598\)](#) in Contrast
- **Service Key:** The [service key \(page 598\)](#) in Contrast
- **Enrichment Excluded Fields:** Fields to exclude from the API response
- **Max Retries:** Max number of retries to be performed in connection errors

4. Select **Submit**.

## View Contrast ADR data in Splunk

Splunk provides three dashboards where you can see Contrast data.

1. In **Splunk**, under **Apps**, find **Contrast ADR for Splunk**.
2. Select **Dashboards**.
3. Select a dashboard:
  - **Attack Dashboard:** This dashboard shows a summary of attacks that Contrast blocked in a specified time frame, identifies which applications are targeted for attacks, the type of attacks and when they occurred., and the most frequently targeted URIs.
  - **Attacks by Applications:** This dashboard shows the number of attacks, the different types of attacks detected and blocked, the distribution of different attack types in a specified time frame, and the top 10 most attacked URIs for the specified applications.
  - **Attacks Geographical Distribution:** This dashboard shows a geographical view of attacks that Contrast Security detected and blocked.

## See also

- [Attack events \(page 578\)](#)
- [Attacks \(page 584\)](#)

## Integrate with Splunk on-call (formerly VictorOps)

Set up an integration with VictorOps incident management to receive attack notifications from Contrast.

To connect VictorOps:

1. In the **user menu**, go to **Organization settings > Integrations**.
2. Select **Connect** in the VictorOps row.
3. Enter a **Name** for the integration. This is displayed in notifications from Contrast.
4. Use the dropdown to choose the **Message type** of the alert. The default selection is "Critical." For more information about message types, see the VictorOps documentation on [incident fields](#).
5. Enter the **URL** for the connection. You can generate the URL in VictorOps through a REST API endpoint. To get a URL or more information, see the VictorOps documentation on [REST endpoint](#).
6. Select **Test connection**. This process may take a few minutes, depending on the number of your VictorOps projects. The test verifies that Contrast can reach the VictorOps instance and that the specified user can log in.
7. Once a connection is made, click in the multiselect field to choose the **Applications** for which you want to send notifications. The default selection is "All Applications."



### IMPORTANT

Contrast will disconnect this integration if it fails to return a successful response after 5 attempts.

## Contrast Visual Studio plugin

Use the Visual Studio plugin to see vulnerability information for instrumented applications from the Visual Studio IDE.

The plugin directs you to a line of code inside Visual Studio, and you can view related details in the Contrast application. This way developers can get application security feedback at the time of development for faster remediation.

The plugin supports Visual Studio versions 2017 (15.0 and later), 2019, and 2022.

To install, configure and use the Visual Studio plugin:

1. In Visual Studio, select **Extensions**.
2. In the new window, select **Online** from the left navigation panel.
3. Search for "Contrast", and select **Download**.
4. After you finish the download, restart the IDE.
5. In Visual Studio, go to **Tools > Options**.
6. In the search, enter "Contrast Security" and select **Contrast Security - Connection**.
7. In the **Contrast Connection** form, add the **Contrast URL**, **Username**, **Service key**, **API key** and **Organization ID** in the appropriate fields. You can find these in your [profile \(page 598\)](#).



### NOTE

The API key must belong to the organization you want to access or you'll get authorization errors. After many failed attempts, this will lock your account.

8. Select **Add**. Visual Studio automatically tests the connection as it attempts to retrieve the organization from Contrast.
9. Select the organization in the **Organizations** field, and select **OK**.
10. In Visual Studio, go to **View > Other Windows > Contrast Security Integration**. You can also search for "Contrast Security Integration". This view shows a list of all the vulnerabilities from Contrast.
11. To filter the list, click the **Filter** icon at the top-left corner of the page.
12. In the window that appears, choose from multiple filters, including servers, applications, severity levels, states and last detected dates.
13. If you can't see your vulnerabilities list, select **Refresh**. To clear all selected filters, click the Clear icon. This also applies for Server and Application lists.



### NOTE

If you can't see your vulnerabilities even after refreshing the list, you must filter your vulnerabilities. You must repeat this process after selecting a different organization in the **Connection** settings so that filters and vulnerabilities are refreshed correctly.

14. Under the **Actions** column, you can click the magnifying glass icon to see more information about the vulnerability. Use the icon to go to the **Vulnerability** page in Contrast.

## Contrast Visual Studio Code plugin

Use the Visual Studio Code plugin to see vulnerability information for instrumented applications from Visual Studio Code environments when Contrast discovers security problems during functional tests.

The plugin shows you an overview of all vulnerabilities found in the application, as well as details for each vulnerability, like the HTTP request that exposed the vulnerability to Contrast.

The plugin supports Visual Studio Code versions 1.42.1 and later.



## NOTE

Integrations are to be used with [Contrast Assess \(page 28\)](#).

To install, configure and use the Visual Studio Code plugin:

1. In Visual Studio Code, go to the **Extensions** view and search “Contrast Security”.
2. Select **Install**. After installation, restart Visual Studio Code.
3. To authenticate to your Contrast account, select the **Settings** icon in the **Contrast Security** view.
4. Select **Workspace** and enter your **API key**, **Organization ID**, **Contrast URL**, and **Authorization header**. You can find these values in your [profile \(page 598\)](#).
5. Select **Test Contrast connection** to validate your credentials. You will see a message that confirms either a successful connection or invalid credentials.
6. Select the **Refresh** icon to update vulnerability information. Under **Contrast Security**, you can see vulnerabilities grouped by **Severity** and ordered by **Status**. Select a vulnerability to view more details like **How to Fix**, **HTTP Information**, **Details**, and **Overview** . You can also see when the vulnerability was last detected and the current status. Vulnerability details display in the code editor under **Output**.



## TIP

With the plugin, you can filter vulnerabilities by:

- Vulnerability metadata:
  - Application name
  - Status (such as Reported, Not a Problem, Remediated)
  - Environment (development, test, or production)
  - Tags (custom labels applied to vulnerabilities)
  - Detection date (specifically, First and Last detected)
- Session metadata:
  - Committer
  - Commit hash
  - Branch name
  - Git tag
  - Repository
  - Test run
  - Version
  - Build number

For example, you can choose to display only those vulnerabilities found on a specific feature branch (Branch Name) and committed directly by you (Committer), filtering out vulnerabilities introduced by a different developer on a separate feature branch.

Someone else can choose to filter vulnerabilities so that they only see results from a specific build (Build Number) that was blocked by their security team. They can immediately pinpoint the subset of vulnerabilities that need to be resolved before deploying the merged feature branch.

## Contrast Visual Studio for Mac plugin

Use the Visual Studio for Mac plugin to see vulnerability information for instrumented applications from the Visual Studio for Mac IDE.

The plugin directs you to the line of code and you can view more details in the Contrast Security pad. This way, developers can get application security feedback at the time of development for faster remediation.

The plugin supports Visual Studio for Mac versions 8.3.0 and later.

To install, configure and use the Visual Studio for Mac plugin:

1. From the [Contrast distribution repository](#), download the Visual Studio for Mac plugin file (.mpack).
2. In Visual Studio for Mac, go to **Visual Studio > Extensions**.
3. Click **Install from file...** and select the downloaded .mpack file. Allow the plugin to install, and restart Visual Studio for Mac.
4. Select **View > Pads > Contrast**. Then select **Configure** and add your **Contrast URL**, **Username**, **Service key**, **API key** and **Organization ID** in the given fields. You can find these in your [profile \(page 598\)](#).
5. Select **Test connection** to test your connection with Contrast. If the connection is successful, select **Save**.
6. Once the plugin is installed, you can return to **View > Pads > Contrast**, select the magnifying glass icon and then select an application. This will load vulnerabilities that Contrast has found for the application. Select **Refresh** if you don't see the vulnerabilities list. Under each section, vulnerabilities display in order by severity, then by status.

You can select a vulnerability to see general or more detailed information. The **General Information** section includes severity, application, status, and history. The **Details** section includes information pulled from Contrast, such as details, notes, and activity for that vulnerability.

To clear all selected filters, select the broom icon. You can also view **Server** and **Application** lists in this way.

## Integrate Wiz with Contrast

Integrate Contrast with Wiz to send an application's runtime security information to a Wiz deployment.

Polling for Assess vulnerabilities occurs once every 24 hours.

### Before you begin

- This integration is supported for hosted customers only.
- Create a service account in Wiz and copy the Client ID and Client secret as described in [Turn on the Contrast integration in Wiz \(page 1117\)](#).
- These cloud providers and instance types are supported:
  - AWS - ARN  
[Access instance metadata for an EC2 instance - Amazon Elastic Compute Cloud](#)
  - Azure - resourceId  
[Azure Instance Metadata Service for virtual machines - Azure Virtual Machines](#)
- Verify that you are using the minimum supported Contrast agent version:
  - Java 6.8.0
  - .NET Framework 51.0.3
  - .NET Core 4.2.19
  - Node.js 5.12.0
  - PHP 1.35.0



- Python 8.5.0
- Go 6.9.0

## Turn on the Contrast integration in Wiz

1. In the Wiz user interface, go to **Settings > integrations**.
2. Select **Add Integration**.
3. Search for **Contrast**.
4. Select the settings you want and select **Add Integration**.
5. Copy the Service Account Token data for the Contrast configuration.

## Connect to Wiz

1. In Contrast, go to the user menu and select **Organization settings > Integrations**.
2. Under Platform integrations, select **Wiz**.
3. In the Credentials tab, specify Wiz credentials:  
These credentials connect Contrast with Wiz:
  - a. Enter these details:
    - **Client ID**: The Client ID generated when you create a Wiz service account.
    - **Client secret**: The Client secret generated when you create a Wiz service account.
    - **Token URL**: Locate the Token URL in the settings page for the Wiz service account.
    - **Tenant Data Center**: Locate your Tenant Data Center value in the Data Center and Regions tab in the **User Avatar** (upper-right-hand corner) > **Tenant Info** page.
  - b. Select **Test connection** to verify the credentials are correct.
  - c. Select **Save**.
4. Configure integration settings.  
These settings determine the type of information Contrast sends to Wiz:
  - a. In the Configuration tab, select any of these options:
    - **Assess (IAST)**: This option sends vulnerability information from Contrast Assess.
    - **Protect (RASP)**: This option sends attack information from Contrast Protect.  
*This option is reserved for future use.*
  - b. Select **Save**.
  - c. Verify that the configuration shows these details:
    - **Created at**: The date you created the integration configuration.
    - **Category**: Application Security
    - **ID**: The Client ID
    - **Behavior**: Pull, Enrich
    - **Status**: Active

# Administration

Users with different [roles and permissions \(page 1264\)](#) have different levels of access in Contrast. This allows different individuals and teams in your business to best use Contrast for their responsibilities:

- [Rules and policy administration: \(page 1118\)](#) RulesAdmins can create and edit policies. Editors can add applications and manage some content details like scores and notifications.
- [Organization administration: \(page 1153\)](#) Organization Administrators can configure settings for a particular organization.
- [System administration: \(page 1185\)](#) For on-premises customers, SuperAdmins can install, configure and maintain Contrast at a system level. ServerAdmins and System Administrators can also help with this responsibility.

## Rules and policy administration

Maintaining your applications in Contrast requires different roles and permissions depending on what you'd like to do.

As a RulesAdmin, select **Policy management** in the **user menu**.

Rule	Severity	Description	Development	QA	Production
Anti-Caching Controls Missing .NET Framework, .NET Core, Go, Java, Node, Python, Ruby	NOTE	Verifies that caching controls are used to protect application content.	168	168	180
Application Disables "secure" Flag on Cookies Java	MEDIUM	Verifies that cookies have the 'secure' flag.	71	71	71
Arbitrary Server Side Forwards .NET Framework, Java	HIGH	Verifies that no untrusted data is used to build a path used in forwards.	146	146	146
Authorization Rules Misordered .NET Framework	MEDIUM	Verifies that the application's authorization rules do not include an allow all user rule before a deny rule.	75	75	75
Authorization Rules Missing Deny Rule .NET Framework	MEDIUM	Verifies that the application's authorization rules include a deny rule.	75	75	75
Cache Control Header Disabled .NET Framework	MEDIUM	Verifies that the application does not disable the cache control header which helps prevent disclosure of sensitive information via the browser cache.	75	75	75
Cookie Has No "secure" Flag .NET Framework, .NET Core	MEDIUM	Verifies that cookies have the 'secure' flag.	77	77	77
Cross-Site Request Forgery Java	HIGH	Verifies that tokens are used to prevent forgeable HTTP traffic	71	71	71
Cross-Site Scripting All application languages	MEDIUM	Verifies that no untrusted data is used in generated HTML pages	181	181	181
Detailed Error Messages Displayed .NET Framework	MEDIUM	Verifies that the application does not inadvertently reveal sensitive or technical information to an attacker via detailed error messages.	75	75	75
Event Validation Disabled .NET Framework	MEDIUM	Verifies that the application does not disable ASP.NET event validation.	75	75	75
Expired Session IDs Not Regenerated .NET Framework	LOW	Verifies that the application generates new session IDs rather than allowing attackers to use expired session IDs	75	75	75
Expression Language Injection Java	HIGH	Verifies that untrusted data is not used in the evaluation of JSP Expression Language.	71	71	71
Forms Auth Protection Mode .NET Framework	MEDIUM	Verifies that the application is using both encryption and validation for forms authentication.	75	75	75
Forms Authentication Cross-App Redirect .NET Framework	MEDIUM	Verifies that the application does not allow for cross-app redirects for authentication which can expose the forms authentication ticket in the URL.	75	75	75
Forms Authentication SSL .NET Framework	MEDIUM	Verifies that forms authentication requests must be submitted over SSL.	75	75	75

Here you can manage:

- [Assess rules \(page 1119\)](#)
- [Security controls \(page 1120\)](#)
- [Vulnerability policy \(page 1124\)](#)

- [Set Protect rules \(page 1131\)](#)
- [CVE shields \(page 1134\)](#)
- [Virtual patches \(page 1138\)](#)
- [Log enhancers \(page 1140\)](#)
- [Block or allow IP addresses \(page 1149\)](#)
- [Edit IP source names \(page 1150\)](#)
- [Application exclusions \(page 1142\)](#)
- [Compliance policy \(page 1147\)](#)
- [Library policy \(page 1151\)](#)
- [Sensitive data \(page 1152\)](#)

A user with RulesAdmin permissions can also:

- [Instrument an application \(page 52\)](#)
- [Enable Protect \(page 1155\)](#)
- [Approve or deny pending vulnerability status changes \(page 1038\)](#)
- [Enable notifications \(page 1153\)](#)
- [Set default scoring \(page 1177\)](#)

## Set Assess rules

To view a list of all rules applied, select **Applications > Your application name > Policy > Assess** or under the user menu, select **Policy management > Assess rules**. Each rule is listed with a severity and description, as well as an indicator of which environments it applies to.

You can also [set the default Assess rules for an organization \(page 1120\)](#).

## Before you begin

- Ensure that you have an Organization Administrator or RulesAdmin role.
- Log in and select the correct organization.

## Steps

Apply Assess rules and settings:

1. To apply Assess rules to particular environments for applications:
  - a. When viewing the list of rules under **Applications**, use the toggles to turn each rule on or off for each environment. You can also use the checkboxes in the left column to select multiple rules, then select **Change Mode** to apply them. In the window that appears, toggle the rules on or off for each environment and select **Done**.
  - b. Alternatively, under **Policy management > Assess rules**, select a rule to see a list of applications that are associated with that rule. Use the toggles to turn rules on or off for each application.
2. To update settings for individual Assess rules:
  - a. Under **Policy management**, select the name of rule to show a list of applications associated with the rule.
  - b. To select one or more applications, select the check box next to each application. To select all applications, select the **Application** check box.
  - c. Select the **settings** icon (⚙️) in the top right.
  - d. In the window that appears, select the **Likelihood**, **Impact** and **Confidence Level** of the vulnerabilities for which this rule is intended.
  - e. Optionally, select the checkbox to **Override** to enable this option to update these fields after the configuration is saved.

- f. In the **Risk Description** field, enter additional information regarding potential consequences of exposure to this vulnerability. You can also provide a **Recommendation**
- g. In the **References** field, enter a link to an external reference related to the specific vulnerability to provide more context for the rule.
- h. Select **Save**.

## Set Assess rules for organizations

When you add and configure an agent for an application or create a new organization, Contrast applies a set of default Assess rules.

Use this procedure to change the default settings for Assess rules at an organization level. These settings apply to any new application that you add to a Contrast organization. These changes have no affect on existing applications in the organization.

### Before you begin

- Ensure that you have an Organization Administrator or Organization RulesAdmin role.
- Log in and select the correct organization.

### Steps

1. Under the user menu, select **Policy management**.
2. Select **Configure the default policy**.

3. Select **All** as the filter.

4. For each Assess rule, use the toggles to turn each rule on or off for each environment.

Rule	Severity	Description	Development	QA	Production
Anti-Caching Controls Missing All application languages	NOTE	Verifies that caching controls are used to protect application content.			

## Security controls

A security control is a method in your code that ensures the data passing through it is safe to use in your application. Contrast trusts information that is passed through built-in security controls it knows about. Contrast monitors many methods from third-party libraries to determine whether a data flow is safe.

In some cases, your organization may choose to build its own security controls which are not known to Contrast. For these cases, create security controls that teach Contrast to account for your custom method. Adding custom security controls results in Contrast reporting more accurate results.

## Types of security controls

- **Input validators:** Validators are methods that ensure only properly formed and formatted data is accepted as input before it's passed to other parts of the application. They are designed to allow the input field to accept or reject specific characters.  
Input validation is the primary method of preventing SQLi, XSS, and other input-validation related attacks.
- **Regex validators:** Regex validators compare a specified regex pattern in an input string to validate whether it's safe. If a string matches the regular expression specified in a security control, Contrast suppresses related vulnerabilities. Regex validators are applied on an application-specific basis.
- **Sanitizers:** Instead of validating input, sanitizers render input safe before it's passed to other parts of the application, like databases. For example, a sanitizer might take a single quote that could be used in a potential injection attack and change it to double quotes.

## When to use security controls

Create security controls when Contrast does not have visibility into methods, classes, or libraries that your application is using to protect it from input validation issues (sanitizers, input validators, and regex validators).

If you know about the validators or sanitizers that your applications use, you can add them manually or move a suggested security control that Contrast detects automatically to the list of security controls that you apply to your applications.

After you add and enable security controls, they suppress input vulnerabilities in Assess that the security control is designed to mitigate. It is very important that you apply your security control to the proper vulnerability rule.

## Effects of using security controls

Security controls affect any vulnerability and detection for specified rules. Input validators and sanitizers are usually designed for or are applicable to a specific type of data, a particular field, page, or specific application. Enabling a security control for all rules can result in false negatives findings.

Use security controls carefully. You might need to apply a security control to specific rules only. For example, if a validator or sanitizer protects your application from XSS, it may not be effective against SQL injection. If you apply a security control to all rules, Contrast would likely suppress aSQLi vulnerability which would results in false negative finding.

A security control should be good enough to assure you that is it protecting your application against a wide range of attacks.

## Roles for security controls management

Only users assigned an organizational role of RulesAdmin or higher can view or modify security controls.

## Supported languages

Security controls for input validators and sanitizers apply to Java, .NET Framework, and .NET Core languages only.

Security controls for regex validators apply to the Java language only.

## Security control example

Assume that you have a method called `DoLegacySecurity()` inside a class called `com.Acme.OldSecurity` that is being reported for using insecure cryptographic algorithms. You create a security control and specify this method signature:

```
com.Acme.OldSecurity.DoLegacySecurity(java.lang.String*)
```

**Policy Management**

ASSESS

Assess Rules

Security Controls

Vulnerability Management

PROTECT

Protect Rules

CVE Shields

Virtual Patches

Log Enhancers

IP Management

GENERAL

Application Exclusions

Compliance Policy

Library Policy

Sensitive Data

**Add Security Control**

Name

Insecure cryptographic algorithm

Language

Java

Type

Input Validator

API

com.Acme.OldSecurity.DoLegacySecurity(java.lang.String\*)

Applicable rules

Unvalidated Redirect

Cancel Add

In this example, `Java.lang.String*` is the marked parameter to be validated.

When creating the security control, you are careful to not include any trailing parameter definitions or extra characters.

Contrast matches this method signature against the stack trace for any vulnerabilities it finds and suppresses them.

## Add, edit or delete security controls

### Before you begin

- Security controls for input validators and sanitizers apply to Java, .NET Framework, and .NET Core languages only.
- Security controls for regex validators apply to the Java language only.

### Steps


1. Select **User Menu > Policy Management**, select **Security controls**.  
The Security Controls grid shows a list existing security controls, if there are any.
2. Select the name of an existing security control to edit, or select **Add security control** to create one.
3. In the panel that opens, specify this information:
  - **Name**
  - **Language**: Select Java, .NET Framework, or .NET Core.
  - **Type**: Select one of these methods:
    - **Input validator** accepts user input and take corrective action if unsafe data is received.
    - **Sanitizer** cleans the data that is passed in, making it safe for consumption by any interpreter. Many sanitizers prevent one type of attack, but not another.
    - **Regex validator** compares a specified regex pattern in an input string to validate whether it's safe.
  - For **input validators or sanitizers**, specify the API to use.  
When specifying the API for input validators and sanitizers, consider these conventions:

- **Java:**  
Java must include a method name and parameters. Use fully qualified types, intended to target only `java.lang.String` parameters (not boolean, int, long, short double, float, and so forth).
  - **.NET Framework and .NET Core:**
    - Include a return type (or void), method name and parameters. Use fully qualified types, intended to target only `System.String` parameters.
    - Verify that no white space exists between the parameters.
    - Mark the parameters that are going to be validated or sanitized with an asterisk ( \* ).
  - For **Regex validators**, specify this information:
    - **Regex pattern:** Specify the regex pattern you want to compare with an input string. The [Regular expression references \(page 1123\)](#) provides examples of patterns you can use.
    - **Application:** Specify the applications where you want to apply the regex validator.
  - **Applicable vulnerability rules:** Choose **All**, or select one or more individual vulnerabilities.
4. Select **Save** to create a new security control. If you are editing an existing security control, you also have the option to delete the security control from this panel with the **Delete** icon.
  5. At the bottom of the table, you will see **Suggestions** for potential security controls that Contrast detects, along with their class and method. (You can hide the section by clicking on the caret in the header row.)

If a security control is automatically discovered for the first time, a notification is sent to all users with at least View permissions for the corresponding applications.

Hover over the API to see where this suggestion was discovered, and optionally, select the name of the application to see the vulnerabilities in context of that application.

Use the **plus** icon ( + ) at the end of the suggestion row, to add the suggestion as a new security control and include it in the table above. You can edit the Name, API and Type fields inline before adding it. After you add the security control, select the name and verify that the security control is applied to the correct application rules.

Use the **Delete** icon (  ) to delete the suggestion. Contrast doesn't repeat suggestions, so once you delete it, an API is never suggested again. There is no way to view historical suggestions or get them back.



## NOTE

Servers may require restart. Contrast provides a list of servers affected by your selection.

## Create security controls for specific vulnerabilities

You can also create security controls in the context of a particular vulnerability with a tag event.

If Contrast has captured runtime data flow for a vulnerability, you can select **Vulnerabilities > Vulnerability name > Details** to see more information about that vulnerability. Potential security controls that are detected trigger a tag event and this is shown as a low severity (green) event. Expand the event and you can select **Add a security control**.

Also, if you mark a vulnerability as **Not A Problem** with the reason "Goes through an internal security control," you can define that security control at that time.

## Regular expression reference

Use this table, and the examples below, for reference when [creating application exclusions \(page 1142\)](#):

Effect	Pattern	Example pattern	Example match
Start of a string	<code>^</code>	<code>^w+</code>	Start of a string
End of a string	<code>\$</code>	<code>w+\$</code>	End of a string
Case-insensitive match of following string	<code>(?i)</code>	<code>(?i)%0a</code>	%0a or %0A
A single character of: a, b or c	<code>[abc]</code>	<code>[abc] +</code>	a bb ccc
A character except: a, b or c	<code>[^abc]</code>	<code>[^abc] +</code>	Anythingbutabc.
A character in the range: a-z	<code>[a-z]</code>	<code>[a-z] +</code>	Only a-z
A character not in the range: a-z	<code>[^a-z]</code>	<code>[^a-z] +</code>	Anythingbuta-z.
A character in the range of: a-z or A-Z	<code>[a-zA-Z]</code>	<code>[a-zA-Z] +</code>	abc123DEF
Any single character	<code>.</code>	<code>. +</code>	abc
Any whitespace character	<code>\s</code>	<code>\s</code>	anywhitespacecharacter
Any non-whitespace character	<code>\S</code>	<code>\S +</code>	any non-whitespace
Any digit	<code>\d</code>	<code>\d</code>	not 1 not 2
Any non-digit	<code>\D</code>	<code>\D +</code>	not 1 not 2
Zero or one of a	<code>a?</code>	<code>ba?</code>	ba b a
Zero or more of a	<code>a*</code>	<code>ba*</code>	a ba baa aaa ba b
One or more of a	<code>a+</code>	<code>a+</code>	a aa aaa aaaa bab baab
Exactly 3 of a	<code>a{3}</code>	<code>a{3}</code>	a aa aaa aaaa
3 or more of a	<code>a{3,}</code>	<code>a{3,}</code>	a aa aaa aaaa aaaaaa
Between 3 and 6 of a	<code>a{3,6}</code>	<code>a{3,6}</code>	a aa aaa aaaa aaaaaa aaaa
Period (dot) is a literal character	<code>.</code>	<code>a.b</code>	string.string

## Vulnerability management policies

Vulnerability policies let administrators with Organization RulesAdmin or Organization Administrator roles define a set of criteria that, when triggered, either changes the status of a vulnerability or flags it for review. The criteria that define the policy includes vulnerability rules, severity, application, and route.

- If you are using organization users and groups, you need an Organization RulesAdmin or Organization Administrator role.
- If you are using role-based access control, you need a role with the Organization Administrator role.

Vulnerability policies result in a more accurate view of which vulnerabilities need attention and which are considered remediated and closed. You have the option of setting vulnerability behavior, auto-verification policies, violation policies, and time to remediate settings.

You can set [in-app notifications \(page 1175\)](#) when vulnerabilities violate these policies. Administrators are notified of violations in-app and by email.

## Vulnerability behavior

The vulnerability behavior settings let you determine if administrator approval is required when users try to close vulnerabilities. You can also choose to display a custom label for the reason that's different from Other when users change a vulnerability status to Not a Problem.

## Types of Auto-verification policies

[Auto-verification \(page 1126\)](#) policies automatically change the status of a vulnerability that meets specific criteria to **Remediated - Auto-verified**. These policies are useful when you want a more accurate view of vulnerabilities that need attention instead of relying on manually changing the status of vulnerabilities after they are fixed and verified.

An auto-verification policy can be session-based, route-based, or time-based:

- **Session-based auto-verification (recommended):** A combination of metadata values that you set in the agent configuration file define a session. You control when a session ends by calling a Contrast API at the end of a test run.



You need an automated test suite for this type of auto-verification.

- **Route-based auto-verification:** A combination of metadata values that you set in the agent configuration file define a session. Use this type of auto-verification if you cannot use session-based auto-verification.

You need an automated test suite for this type of auto-verification.

- **Time-based auto-verification:** Use this method of auto-verification if you have high confidence that your application will exercise all routes within a specified time period.

You can use automated or manual testing with this type of auto-verification.

## Auto-verification behavior

- Contrast marks a vulnerability as **Remediated - Auto-Verified** if Contrast does not discover it on the same route across two different sessions. If two sessions report the exact same session metadata values, Contrast views the two sessions as a single session.

Depending on the defined values, each agent run could be part of a single session or every agent run could have its own session. If you are integrating Contrast into a CI/CD pipeline, ensure that you send at least one session metadata key-value pair that is unique each time you deploy a new version of the application. For example, configure the agent to send the Commit Hash, Build Number, or Version metadata because these values are likely to change for each application deployment. When you use route-based auto-verification, Contrast evaluates vulnerabilities associated with the route immediately after the route is exercised. Alternatively, when you use session-based auto-verification, Contrast waits to evaluate all routes exercised during the session until after calling the Session End API.

Contrast closes vulnerabilities only if the vulnerable parameter that the application reads during the observation of an exercised route vulnerability is not reported again. If the route is exercised, but Contrast does not see the particular parameters that are identified in the vulnerability as being a source of tainted data, then the vulnerability remains open.

- If a vulnerability that Contrast previously marked as **Remediated - Auto-Verified** reappears when the same route is exercised, its status changes to **Reported**. Contrast updates the details in the Activity tab on the vulnerability details page.
- If a vulnerability that Contrast previously marked as **Remediated - Auto-Verified** reappears when the same route is exercised after you disable or delete an auto-verification policy, the vulnerability status changes to **Reported**. Contrast updates the details in the Activity tab on the vulnerability details page.

## Session metadata for session-based and route-based auto-verification

For session-based or route-based vulnerability policies, [add session metadata \(page 618\)](#) to the agent configuration files:

- Providing unique session metadata allows Contrast to create a baseline of findings that lets it verify whether a vulnerability was remediated based on route comparisons.
- Using the Test Run session metadata field is a good way to ensure that Contrast is tracking routes and vulnerabilities across an entire test run even if you restart the agent and the application multiple times during the run.

Contrast creates a unique session ID for every unique metadata key-value pair. Using session metadata in this manner combines multiple test runs into a single test session. This action is useful in situations where different code paths on the same route are tested.

- Using the Commit Hash, Build Number, or Version metadata is useful because these values are likely to change for each application deployment.

## Violation policies

[Violation policies \(page 1130\)](#) trigger a violation notice when a vulnerability matches a set of specific criteria. If triggered, you see the vulnerability in red text in the vulnerabilities list. Use the vulnerabilities filter to view only vulnerabilities with policy violations.

CRITICAL	SQL Injection from "orderBy" Parameter on "/ggtmtnOUDb/okTheZlhlv" page	<b>Attention</b> This vulnerability is in violation of the following policies set by your administrator All must be remediated in 28 days	14
----------	-------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------	----

## Policy triggers

These trigger types activate a vulnerability policy:

- **Session-based (recommended):** Triggers an auto-verification policy when a vulnerability is seen, or not seen, on a specific route during a session. When you use this trigger, use the Contrast API to end the session. This feature lets you define when a session ends so that you can get immediate results from a test run.
- **Route-based:** Triggers an auto-verification policy when a vulnerability is seen, or not seen, on a specific route. This trigger is available for technologies where Contrast can identify routes.
- **Time:** Triggers a violation or auto-verification policy after a specified number of days.

## Environments

For optimal results, configure the vulnerability policies to apply to the environments where you are using test automation. If you are running the same application on multiple servers, ensure that each server is configured for the Development, QA, or Production environment.

## Multiple policy actions

If multiple policies affect the same vulnerability, these rules determine how Contrast applies the policies:

- Auto-verification policies take precedence over violation policies. For example, if an auto-verification deadline applies first, the vulnerability is closed and never flagged.
- Between two time-based triggers, the action with the closest deadline applies first. For example, if a violation deadline applies first, the vulnerability is flagged and then auto-verified when the later deadline applies.

## Time to remediate

The time to remediate settings let you specify the default filter for the Average time to remediate section in the [Contrast dashboard \(page 596\)](#).



The Contrast dashboard presents data on the Average Time to Remediate (ATTR) vulnerabilities at your organization. When generating these metrics, you can choose to filter on the date a vulnerability was first seen, last seen, or closed, depending on your organizations program goals.

Based on the selected time frame, Contrast finds vulnerabilities that are closed and have a date that falls within the filter range. Contrast calculates the ATTR by looking at the difference between the first seen and closed dates on the selected vulnerabilities, and averaging them.

## Set auto-verification vulnerability policies

[Auto-verification policies for vulnerabilities \(page 1124\)](#) automatically change the status of a vulnerability that meets specific criteria to **Remediated - Auto-verified**. These policies can be session-based, route-based or time-based.

When you add a policy, it is turned on, by default. You can turn a policy off or on in the Enabled column in the Auto-verification tab.

Auto-Verification		Violation			
Find Policy					+ Add Policy
Policy	Vulnerability Rules	Applications	Description	Environments	Enabled
test	Note	All		All	
Simon Test Time-Based	All	WebGoat7	7 day duration	All	

## Before you begin

- Use Contrast version: 3.7.2 and later.
- Verify you are using the minimum version of supported agents:
  - .NET Framework 20.4.1
  - .NET Core 1.0
  - Java 3.7.3.14895
  - Node.js 2.11.0
  - Python 3.4.0
  - Ruby 3.8.4
- Verify that you are using a [supported framework](#). (page 1129)
- If you plan to use session-based or route-based auto-verification, [configure unique session metadata](#) (page 618) (for example, Commit Hash, Build Number, or Version) in the agent configuration files.

## Set an auto-verification policy

1. From the user menu, select **Policy management**.
2. Select **Vulnerability management**.
3. In Vulnerability policy, select the **Auto-verification** tab.
4. Select **Add policy**.
5. In **Name**, enter a name for the policy.
6. In **Vulnerability rules**, select one or more severity levels or Assess rules that you want to associate with the policy.
7. In **Applications**, select one or more importance levels or applications that you want to associate with the policy.  
To find specific levels or applications, select the Applications box and start typing.
8. In **Environments**, select one or more server environment where the policy is applied: All environments, Development, QA, or Production.
9. Under **Trigger**, select the type of trigger you want to use for the policy (select one or both types of triggers):
  - To set a time-based trigger, select **Mark any vulnerability as “Verified – Auto-Remediated” after** and select the number of days after which the vulnerability policy is marked as auto-verified.  
This trigger is useful if you are confident that vulnerabilities will be fixed and routes will be exercised within the selected time frame. If Contrast finds the vulnerabilities again, it reopens them.

**TIP**

Use time-based auto-verification along with session-based or route-based auto-verification to find situations where routes change drastically from build to build. For example:

- Major code refactoring where you add new routes and remove old routes.
- A route is no longer exercised because it is no longer valid.

- To set a trigger for session-based or route-based auto-verification:
  - a. Select **Auto-verify based on session or route**.  
Session or route-based auto-verification take precedence over a time-based trigger.
  - b. Select **ONE** of these options (you can't use both options at the same time):
    - **Session-based auto-verification (recommended)**: This type of auto-verification policy lets you define when a session ends so that you can get immediate results from a test run, including making pass/fail decisions for your builds. Session-based auto-verification is the preferred method for auto-verification. When you select this option, you end sessions by adding calling the Contrast API at the end of a test run.
    - **Route-based auto-verification**: If you can't use session-based auto-verification, consider using route-based auto-verification. In this case, the session metadata that you configure for your agent define the session.  
Route-based triggers only work for certain technologies with identifiable routes.

10. Select **Save**.

### Configure a test run for session-based auto-verification

Session-based auto-verification, the recommended auto-verification method, requires you to make calls to the Contrast SBAVRouteSession API at the end of a test run. The following examples show different approaches for closing sessions.

To find your authorization header and API key, log in to the Contrast web interface and under the user menu, select **User settings**.

- End the session defined by a session ID and application ID. Use commands similar to the following example:

```
curl --location --request POST 'https://<HOST>/Contrast/api/ng/organizations/<ORG-UUID>/agent-sessions/sbav' \
--header 'Authorization: <Your-Auth-Header-Value>' \
--header 'API-Key: <API-KEY>' \
--header 'Content-Type: application/json' \
--data-raw '{
 "sessionId": "0",
 "appId": "5b4960b3-a111-4f2a-bf24-7367be7c8302"
}'
```

- End the session defined by a session ID, the application name, and the application language. Use commands similar to the following example:

```
curl --location --request POST 'https://<HOST>/Contrast/api/ng/organizations/<ORG-UUID>/agent-sessions/sbav' \
--header 'Authorization: <Your-Auth-Header-Value>' \
--header 'API-Key: <API-KEY>' \
--header 'Content-Type: application/json' \
--data-raw '{
 "sessionId": "0",
 "appName": "FakeRubyApp",
 "language": "ruby"
}'
```

```
"appLanguage": "JAVA"
}'
```

- End the session defined by metadata key-value pairs configured for the application and the application ID. Use commands similar to the following example:

```
curl --location --request POST 'https://<HOST>/Contrast/api/ng/
organizations/<ORG-UUID>/agent-sessions/sbav' \
--header 'Authorization: <Your-Auth-Header-Value>' \
--header 'API-Key: <API-KEY>' \
--header 'Content-Type: application/json' \
--data-raw '{
 "appId": "abc",
 "metadata": [
 { "label": "developer", "value": "carlos" },
 { "label": "repo", "value": "ts" }
]
}'
```

- End the session defined by these metadata key-value pairs configured for the application with application name and language. Use commands similar to the following example:

```
curl --location --request POST 'https://<HOST>/Contrast/api/ng/
organizations/<ORG-UUID>/agent-sessions/sbav' \
--header 'Authorization: <Your-Auth-Header-Value>' \
--header 'API-Key: <API-KEY>' \
--header 'Content-Type: application/json' \
--data-raw '{
 "appName": "FakeJavaApp",
 "appLanguage": "JAVA",
 "metadata": [
 { "label": "developer", "value": "carlos" },
 { "label": "repo", "value": "ts" }
]
}'
```

## Update a vulnerability policy

1. From the user menu, select **Policy management**.
2. Select **Vulnerability management**.
3. In Vulnerability policy, select the **Auto-verification** tab.
4. Select the policy.
5. Update the values, as needed.
6. Select **Update**.

## Auto-verification supported frameworks

These frameworks support auto-verification policies:

- **Java:** Jersey 2, Spring MVC 4, Struts 1, Struts 2, Servlets (beta)
- **.NET Framework:** ASP.NET MVC (versions 4 and 5), WebForms, WebAPI and WCF
- **.NET Core:** ASP.NET Core MVC (versions 2.1, 2.2, 3.0 and 3.1) and ASP.NET Core Razor Pages (versions 2.1, 2.2, 3.0, and 3.1)
- **Node.js:** Express, Hapi 17+, Koa, and Kraken
- **Python:** Django, Pyramid, and Flask
- **Ruby:** Rails and Sinatra

## Set violation vulnerability policies

Violation policies mark a vulnerability as being in violation of a policy. When this policy is triggered, the vulnerability is displayed in red text in the Vulnerabilities list.

When you add a policy, it is turned on, by default. You can turn a policy off or on in the Enabled column in the Violation tab.

Auto-Verification		Violation				
<input type="text" value="Find Policy"/>						<a href="#">+ Add Policy</a>
Policy	Vulnerability Rules	Applications	Description	Environments	Enabled	
All must be remediated in 28 days Time-Based	All	All	28 day duration	All	<input checked="" type="checkbox"/>	
All High must be remediated in o... Time-Based	Critical High	All	7 day duration	All	<input type="checkbox"/>	

## Before you begin

- An Organization Rules Admin or Organization Admin role is required.

## Steps

1. From the user menu, select **Policy management**.
2. Select **Vulnerability management**.
3. Select the **Violation** tab.
4. To add a policy:
  - a. Select **Add policy**.
  - b. In Name, enter a name for the policy.
  - c. In Vulnerability rules, select the vulnerability severity levels or Assess rules that you want to associate with the policy.
  - d. In Applications, select the application importance levels or applications that you want to associate with the policy.
  - e. In Environment, select the environments for the servers hosting the applications where you want to apply the policy.
  - f. Under Trigger, select **Flag any existing vulnerability after** and select the number of days.
  - g. Select **Save**.
5. To update a policy:
  - a. On the Violations tab, select a policy.
  - b. Change any of the policy values.
  - c. Select **Update**.

## Protect rules

Apply Protect rules to monitor or block specific kinds of attacks in application environments. Every rule represents a type of attack that exploits vulnerabilities in either custom code or open-source libraries, such as SQL injection or cross site scripting.

Contrast includes many Protect rules you can use to monitor or block attacks, like these:

- **Command injection:** Carefully crafted inputs can execute tainted operating system level commands.
- **Cross-site scripting:** A web application vulnerability that can allow users to run arbitrary JavaScript in other user's browsers.
- **Expression language injection:** A vulnerability type for many frameworks and custom code that happens when an application mistakenly evaluates user inputs as expression languages like OGNL, SpEL, or JSP EL.
- **Method tampering:** An attack against authentication or authorization systems that have implicit "allow all" settings in their security configuration.
- **Path traversal / Local file include:** A vulnerability that allows users to control which files an application opens and reads.
- **SQL and NoSQL injection:** Carefully crafted inputs to the application that alter SQL or NoSQL queries in order to steal data or execute code.
- **Unsafe file upload:** A vulnerability in the upload process that allows malicious files to bypass upload protections and perform malicious actions. This rule affects files with commonly-used extensions including (but not limited to): SVG, ASP, ASPX, \*SH, JAR, and JAVA. In Monitor mode, this rule reports potentially unsafe file uploads to Contrast. In Block mode, Contrast blocks uploads of these files.
- **Untrusted deserialization:** A web application vulnerability that allows users to pass arbitrary objects to a deserializer and execute remote code.
- **XML external entity processing:** A vulnerability in XML processing that allows users to read, write, and potentially, execute remote code to a file.

## Set Protect rules

You can set [Protect rules \(page 1130\)](#) that monitor or block attacks in your application environments.

When you add new applications, Contrast applies a set of default Protect rules to them. You can change the modes for an organization's [default Protect rules \(page 1133\)](#).

## Before you begin

- Ensure that Contrast (hosted customers) or a SuperAdmin (on-premises customers) [granted Protect permissions \(page 1226\)](#) for the organization.

## Steps

1. Select **Applications** in the header.
2. Select an application name and select **Policy**.
3. Select **Protect**.

Overview

Vulnerabilities

Attacks

Libraries

Route Coverage

Flow Map

Policy

ASSESS

PROTECT

EXCLUSIONS

All (11) Find Rule

Change Mode

<input type="checkbox"/> Rule	Type/Description	Development	QA	Production
<input type="checkbox"/> Padding Oracle	Protect Rule / A padding oracle attack can be used to decrypt cipher text. Monitor mode only.	MONITOR	MONITOR	MONITOR
<input type="checkbox"/> Unsafe File Upload	Protect Rule / A vulnerability in the upload process that allows malicious files to bypass upload protections and perform malicious actions. Consequences can range from complete system takeover to web server defacement.	OFF	OFF	OFF

4. To find a specific rule, enter the rule name in the search box.
5. For each rule, set the mode for each environment:
  - a. Select the dropdown for each environment.
  - b. Select one of these modes:
    - **Off:** This mode disables the rule.

- **Monitor:** The agent identifies and reports attacks.
- **Block:** The agent identifies, reports and blocks attacks.



### IMPORTANT

If an attack matches a rule and the mode for that rule is set to **Block**, the Java, .NET Framework, and .NET Core agents throw an `AttackBlockedException`.

To ensure the application doesn't crash, edit the application to handle the `AttackBlockedException`.

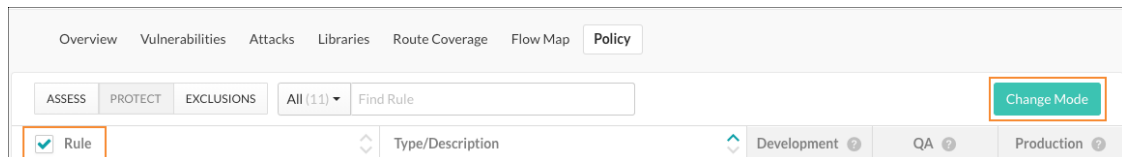
- **Block at perimeter:** The agent blocks a possible attack before the application can process it. This option is not available for all rules.
  - **Monitor at perimeter:** The agent attempts to identify and report a possible attack before the application can process it. This option is not available for all rules.
- If you block or monitor at the perimeter, the agent doesn't verify the attack at the *sink*. This action can lead to false positive results.



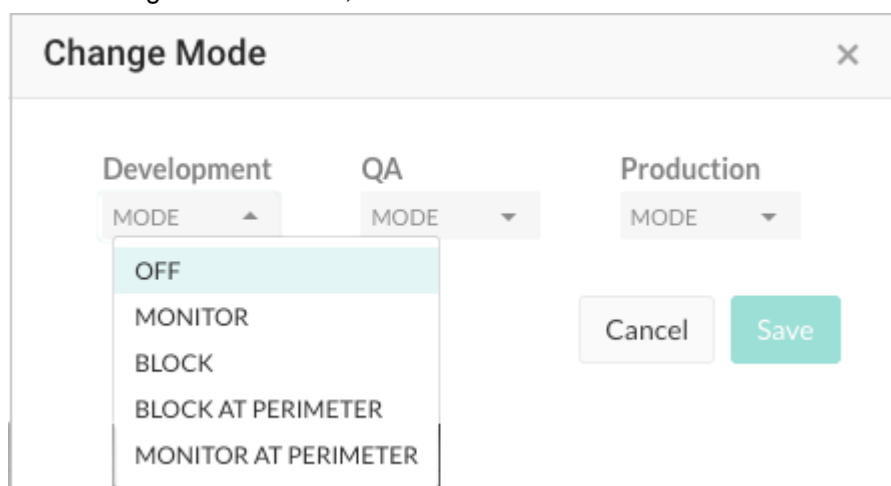
### TIP

You can test policies by setting a different mode for a Protect rule in each environment. This action lets you see how various options work in pre-production and won't disrupt production defenses.

- To apply settings to multiple rules, use one of these methods:
  - Select the checkbox next to each rule that you want to change and select **Change Mode**.
  - To change settings for all rules, select the **Rules** checkbox and select **Change Mode**.



- In the Change Mode window, set the mode for each environment and select **Save**.



- To set Protect rules for all applications in the organization that use a specific rule: This step requires an Organization RulesAdmin role.
  - Select **user menu > Policy management > Protect rules**.
  - To filter the list of rules, use the dropdown to filter the rules by language or the search field to find a rule by name.



- c. Select a rule name to manage settings for all applications that currently use the rule.

**Policy Management**

ASSESS

- Assess Rules
- Security Controls
- Vulnerability Management

PROTECT

- Protect Rules**
- CVE Shields
- Virtual Patches
- Log Enhancers
- IP Management

GENERAL

- Application Exclusions
- Compliance Policy
- Library Policy
- Sensitive Data

PROTECT RULES

OFF MONITOR/MONITOR AT PERIMETER BLOCK BLOCK AT PERIMETER

.NET Framework (15) Find Rule [Configure the default policy](#) for new applications in your organization.

Rule	Description	Development	QA	Production
<b>Command Injection</b> .NET Framework, .NET Core, Java, Node, Python, Ruby	Carefully crafted inputs can execute tainted commands.	4 9	1	
<b>Chained Commands - Command Injection</b> .NET Framework, .NET Core, Node	Detects chained commands as potential attacks since command chaining is often used by attackers to invoke sensitive system commands.	3 1		
<b>Command Backdoors - Command Injection</b> .NET Framework, .NET Core, Node	Detects when user input is executed as a system command or is passed as a command parameter to common shell programs (e.g. /bin/sh -c "some user input").	4		
<b>Dangerous Paths - Command Injection</b> .NET Framework, .NET Core, Node	Detects dangerous paths as part of system commands as attackers often try to get access to sensitive paths or files.	4		

- d. Use the dropdown to set the Protect mode for each environment.

**Policy Management**

ASSESS

- Assess Rules
- Security Controls
- Vulnerability Management

PROTECT

- Protect Rules**
- CVE Shields
- Virtual Patches
- Log Enhancers

**Command Injection**  
Carefully crafted inputs can execute tainted commands.

All (14) Find Application [Change Mode](#)

Application	Importance	Development	QA	Production
<input type="checkbox"/> bugfindy	Medium	BLOCK	BLOCK	BLOCK
<input type="checkbox"/> DonetCoreApp1	Medium	OFF	BLOCK	BLOCK
<input type="checkbox"/> DonetCoreApp2	Medium	BLOCK	BLOCK	BLOCK

## Set Protect rules for organizations

When you add and configure an agent for an application or create a new organization, Contrast applies a set of default Protect rules.



### NOTE

Starting in August 2021, new organizations include an optimized set of Protect rules. This configuration is designed to provide the highest value to users, including enhanced performance.

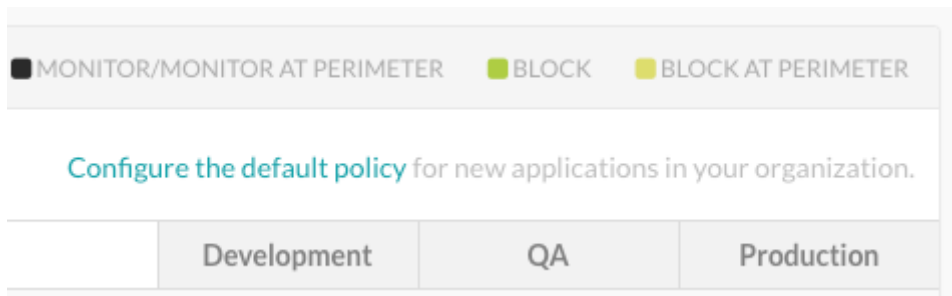
Use this procedure to change the default settings for Protect rules at an organization level. Changing these settings affects new application that you add to a Contrast organization. These changes have no effect on existing applications in the organization.

## Before you begin

- Ensure that you have an Organization Administrator or Organization RulesAdmin role.
- Log in and select the correct organization.

## Steps

1. Under the user menu, select **Policy management**.
2. Select **Protect rules**.
3. Select **Configure the default policy**.



4. For each Protect rule, select the dropdown for the environment where the application is hosted (Development, QA, and Production).
5. Select one of the following modes:
  - **Off:** This mode disables the rule.
  - **Monitor:** The agent identifies and monitors attacks.
  - **Block:** The agent identifies, reports, and blocks attacks.
  - **Block at perimeter:** The agent blocks a possible attack before the application can process it. This option is not available for all rules.
  - **Monitor at perimeter:** The agent attempts to identify and report a possible attack before the application can process it. This option is not available for all rules.

## CVE shields

Common Vulnerabilities and Exposures (CVE) provide a standardized identifier for a given vulnerability or exposure. They also provide a baseline for evaluating the coverage of your tools.

Contrast provides several CVE shields to help protect your applications that contain CVEs. CVE shields are useful for legacy applications that use vulnerable libraries that are difficult to update.

You only need CVE shields when the vulnerability isn't a common attack class like SQL injection or untrusted deserialization. In some cases, Contrast creates a CVE shield to get more data that is specific to a particular threat, even if there's an existing Protect rule that prevents the attack from occurring. This action helps provide more context into exploitation. It helps organizations map ongoing attacks to trends in the overall security ecosystem.

[View CVE shields \(page 1134\)](#)

[Set modes for CVE shields \(page 1136\)](#)

## View CVE shields

The CVE shields list displays the following information:

## Policy management

ASSESS

Assess Rules

Security Controls

Vulnerability Management

PROTECT

Protect Rules

**CVE Shields**

Virtual Patches

Log Enhancers

IP Management

GENERAL

Application Exclusions

CVE SHIELDS

OFF MONITOR/MONITOR AT PERIMETER BLOCK

Find CVE Shield

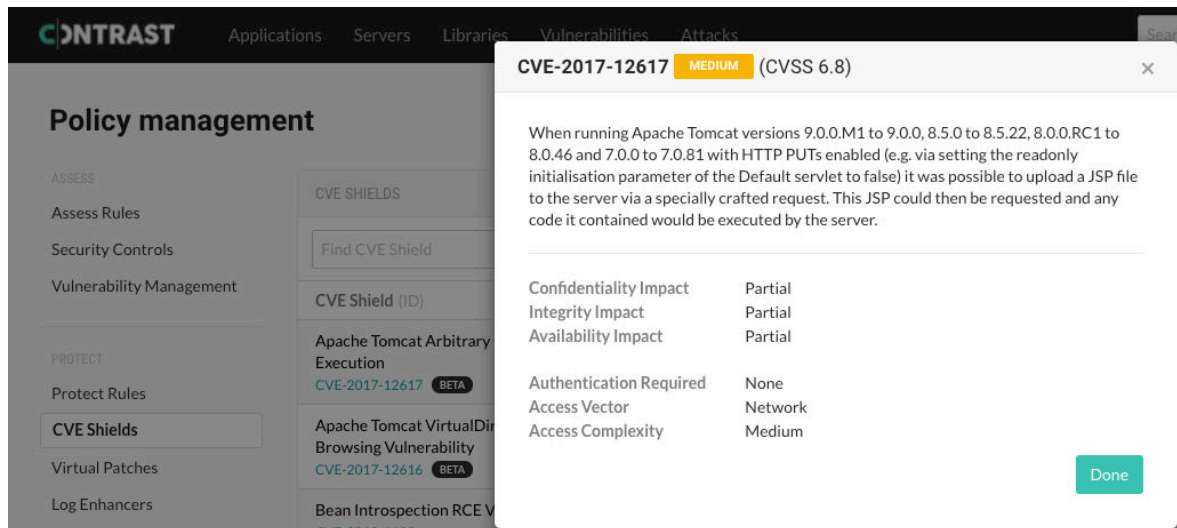
CVE Shield (ID)	Description	Development	QA	Production
<b>Apache Tomcat Arbitrary Code Execution</b> <a href="#">CVE-2017-12617</a> <b>BETA</b>	A remote code execution vulnerability in Tomcat 7.0.0 through 7.0.81, 8.0.0.RC1 through 8.0.46, 8.5.0 through 8.5.22, and 9.0.0.M1 through 9.0.0	1		
<b>Apache Tomcat VirtualDirContext File Browsing Vulnerability</b> <a href="#">CVE-2017-12616</a> <b>BETA</b>	A flaw in VirtualDirContext in Tomcat 7.0.0 through 7.0.80, allows an attacker to view sourcecode and other files on the system.	1		

- The CVE shields that Contrast provides for specific CVEs.
- A description of the CVE.
- The environments in which the servers hosting an application are running.
- The mode configured for the CVE shield:
  - **Off:** This mode disables the CVE shield entirely.
  - **Monitor:** In this mode, the CVE shield identifies and reports attacks.
  - **Monitor at perimeter:** In this mode, the CVE shield tries to identify and report a possible attack before the application can process it. This option is not available for all CVE shields.
  - **Block:** In this mode, the CVE shield identifies, reports, and blocks attacks.
- The applications, if any, that contain a specific CVE.  
The CVE shield defends this vulnerability against attack.

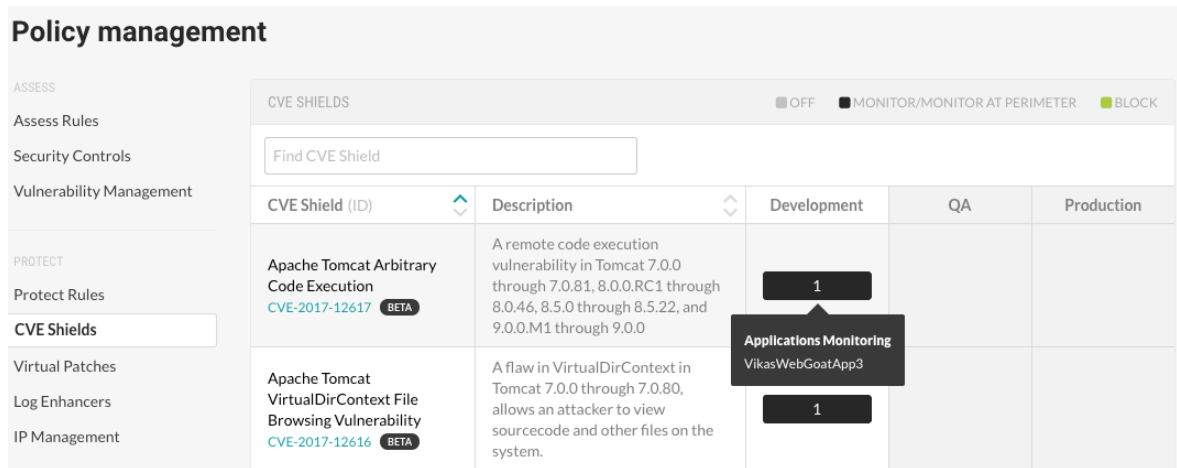
## Steps

To view CVE shields:

1. From the user menu, select **Policy management**.
2. Under Protect, select **CVE shields**.
3. To find a specific CVE, enter a full or partial name in the search box.
4. To view details about a specific CVE, click the link below the CVE name.



5. To view which applications contain a CVE, in one of the environment columns, hover over the number. The tooltip lists the applications that the CVE shield is defending. The number indicates the number of applications that contain the CVE. The mode indicates how the CVE shield is configured.



## Set modes for CVE shields

Instead of detecting categories of attacks, CVE shields defend specific CVEs in applications from attacks.

Set one of the following modes for applications hosted on servers running in a Development, QA, or Production environment:

- **Off:** This mode disables the CVE shield entirely.
- **Monitor:** In this mode, the CVE shield identifies and reports attacks.
- **Monitor at perimeter** In this mode, the CVE shield tries to identify and report a possible attack before the application can process it. This option is not available for all CVE shields.
- **Block:** In this mode, the CVE shield identifies, reports, and blocks attacks.

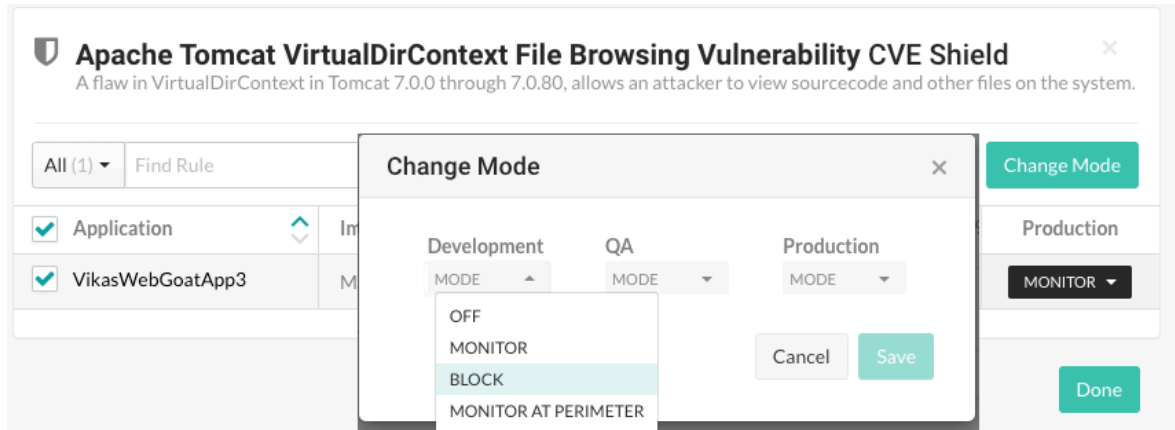
## Before you begin

- **Required:** Check that you have Organization or Rules Admin permissions.
- Check that [settings \(page 915\)](#) for servers hosting your applications are configured to use the correct environments.

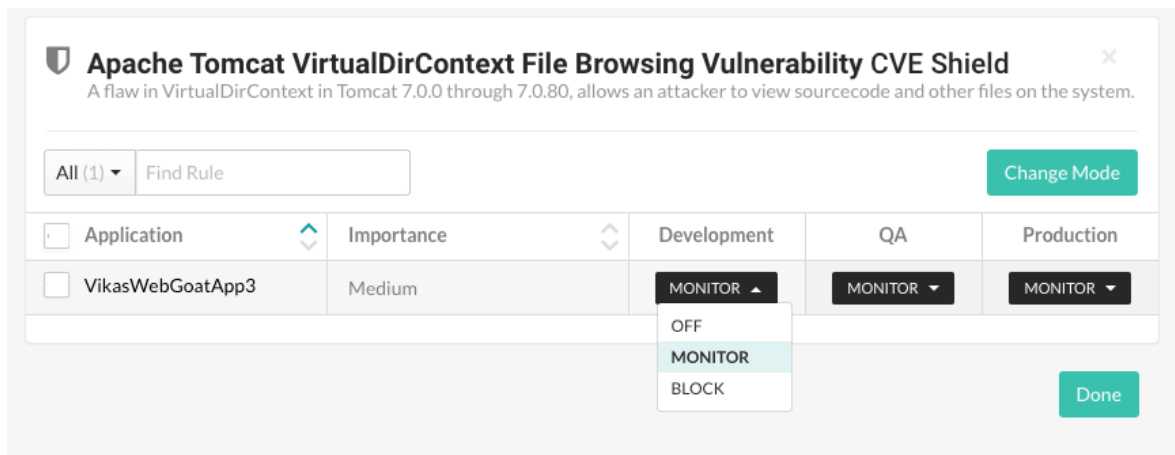
## Steps

To view CVE Shields.

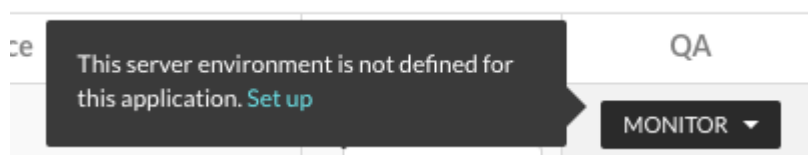
1. From the user menu, select **Policy management**.
2. Select **CVE shields**, and click the name of a CVE shield.  
To find a specific CVE shield, enter a partial or full name in the search box.
3. To set the mode for all or multiple applications:



- To select all applications, select the **Application** checkbox. To select multiple applications, select the checkbox for each application.
  - Click **Change mode**.
  - In the Change Mode window, select the CVE shield mode for the selected applications in a one or more environments.
  - Click **Done**.
4. To set the mode for a single application:



- At the end of the row for an application, select the menu in an environment column.  
If an environment is not defined for the server hosting the application, when you hover on that environment, a tooltip appears. To configure the server for that environment, click **Set up** and select the settings icon (⚙️) at the end of the server row.



- Select a CVE shield mode for the application in the selected environment.
- Click **Done**.

## Set modes for CVE shields for organizations

When you add and configure an agent for an application in a Contrast organization, Contrast applies a set of default CVE shields.



### NOTE

Starting in August 2021, new organizations include an optimized set of CVE shields. This configuration is designed to provide the highest value to users, including enhanced performance.

Use this procedure to change the default settings for CVE shields at an organization level. These settings apply to any new application that you add to a Contrast organization. These changes have no effect on existing applications in the organization.

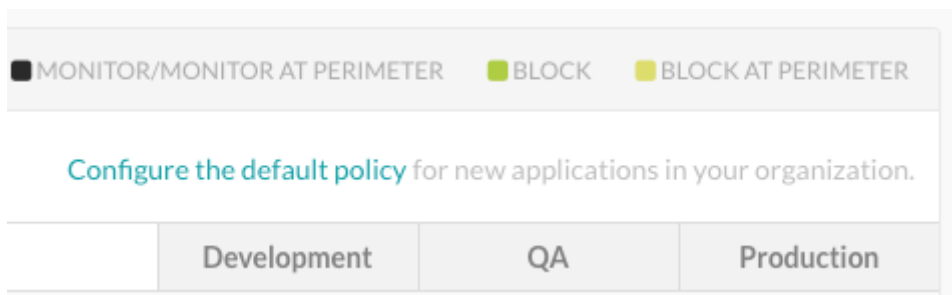
### Before you begin

- Ensure that you have an Organization Administrator or Organization RulesAdmin role.
- Log in to or select the correct organization.

### Steps

To change modes for CVE shields:

1. Under the user menu, select **Policy management**.
2. Select **CVE shields**.
3. Select **Configure the default policy**.



4. For each CVE shield, select the dropdown for the environment where the application is hosted (Development, QA, and Production).
5. Select one of the following modes:
  - **Off:** This mode disables the CVE shield entirely.
  - **Monitor:** In this mode, the CVE shield identifies and reports attacks.
  - **Monitor at perimeter:** In this mode, the CVE shield tries to identify and report a possible attack before the application can process it. This option is not available for all CVE shields.
  - **Block** In this mode, the CVE shield identifies, reports, and blocks attacks.

## Manage virtual patches

Virtual patches are custom, short-term rules that block HTTP requests matching specific criteria (for example, URL, parameter keys or values, and so forth) before an application can process them.

Organization Administrators and RulesAdmins can view and manage virtual patches.

To add a virtual patch:

1. In the **user menu**, under **Policy management**, select **Virtual patches**.
2. Find virtual patches by using the language filters or the search field above the grid.

VIRTUAL PATCHES						
All (4) ▾	Find Virtual Patch		+ Add Virtual Patch			
All (4)	⬆	Description	⬆	Development	QA	Production
Java (4)		Anertix on E				
.NET Framework (0)		Autogenerated patch for Anertix on ErikTomcatEclipse		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
.NET Core (0)						
Node (0)		Blocks Klein for Acct Name		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Ruby (0)						
spring cmd injection		spring cmd injection		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Java						
struts-content-type-header		Struts2 S2-045 Remote Command Execution Patch		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Java						

3. Click on the name of a patch to edit the rule configuration, or select **Add virtual patch** to add a new one.  
You can also select the **Delete** icon to delete a rule or use the toggles in the grid to enable or disable each environment.
4. In the window that appears, add a **Name** and **Description**.

Add Virtual Patch

Name

Name this virtual patch

Description

Describe this virtual patch

Apply To ?

☒ Applications

All applications (155)

☐ Application language

☐ Application technology

Conditions ?

User Agent
matches
Value

User Agent
matches
Value

+ Add Condition

Cancel

Add

5. Under **Apply to**, use the radio button to choose whether the rule applies to specific **Applications**, an **Application language** or an **Application technology**. After clicking the appropriate button, use the multiselect field that appears to further refine your choice.
6. Under **Conditions**, use the dropdowns to select the conditions under which the patch should apply to the applications. Select **Add another condition** in a separate row, if necessary.  
When you select how the virtual patch value is applied, select one of the following options:
  - Equals

- Contains
- Matches (using Perl-Compatible Regular Expressions - PCRE)
- Does not equal
- Does not contain
- Does not match (using Perl-Compatible Regular Expressions - PCRE)

Both the **Matches** and **Does not match** options support the use of Perl-Compatible Regular Expressions (PCRE). If you select **Matches** or **Does not match** option, you can define a regular expression that matches a value in the selected field of the HTTP request.

If the expression matches, or does not match as specified, the virtual patch is applied and the mitigation action specified in the patch configuration is taken.



#### NOTE

Regular expressions can be very powerful, but they can also be complex and difficult to create correctly. If you're not familiar with PCRE expressions, ask for assistance from a security expert or [Contrast Security](#) to ensure that your Virtual Patches are configured correctly and effectively.

As a starting point, look at [Regular expression reference \(page 1123\)](#). This reference provides some examples of what is possible with PCREs.

7. Select **Add** to save the configuration.

## Add or edit log enhancers

Log enhancers are instrumentation instructions that allow the Contrast agent to log additional parameters and data in the application, without requiring any source code changes.

By using these deep security instrumentation techniques, a user can specify the API and parameter to log, and the Contrast agent adds this information to the *security.log* file as part of RASP logging.



#### NOTE

Starting in August 2021, new organizations include an optimized set of log enhancers. This configuration is designed to provide the highest value to users, including enhanced performance.

To add, edit or delete a log enhancer:

1. Under [policy management \(page 1118\)](#), select **Log enhancers**.
2. Filter by language, or use the search to find the existing log enhancer you want to edit and select the name, or select **Add log enhancer**. Use the toggles in the grid row to enable or disable the rule in each environment.



LOG ENHANCERS					
All (18) Find Log Enhancer		+ Add Log Enhancer			
Log Enhancer	Description	Development	QA	Production	
Apache SSL HostName Verifier Changed <small>Java</small>	The SSL hostname verifier changed after the SSLSocketFactory object was created	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
ESAPI Default Login <small>Java</small>	ESAPI login using current request	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
ESAPI Exception <small>Java</small>	ESAPI threw a runtime security exception	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
ESAPI Exception with cause <small>Java</small>	ESAPI threw a runtime security exception	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	

- In the window that appears, enter a **Name** and **Description**.

### Add Log Enhancer

Name

Name this log enhancer

Description

Describe this log enhancer

Log Level

Info

Log Type

Security

API to Log

Language

Java

API

org.springframework.webmvc.DefaultController.onSuc

Format

The user " logged in to the .

Cancel

Add

- Enter a **Log level** (page 1271) and **Log type**.
- Under **API to log**, enter:
  - Language**
  - API:** Use the structure `<class_name>.<method_name>(<argument_types>)`. For example:

```
public boolean com.acme.Authenticator.authenticate(String user, String password)
```
  - Format:** Enter the log description, including relevant data from the function call. You can include any of the following placeholders in your message:
    - `{O}`: Print the string representation of the object on which this call is made. If the method is static, this may be null or empty.
    - `{Pn}`: Print the given parameter at index `n`. Note that `n` starts at 1.
    - `{P1}`: Print the first parameter into the message.
    - `{R}`: Print the return value of the function.
- Select **Add** to save the rule.

## Application exclusions

Exclusions are used to suppress events. You might want to suppress events if you are using an external security control outside of the scope of Contrast's agent instrumentation. For example:

- As an administrator, you need to change the HTML that shows up on your web page, even though this qualifies as a cross-site scripting (XSS) vulnerability. In this case, you can create an exclusion that prevents these changes from being reported.
- You use an edge device to place the correct headers on outbound HTTP responses to stop clickjacking attacks. However, the issue might be appropriately reported because the application never provided the required protection.
- When you test beta rules, you can use exclusions to suppress false positives.

If you are using Java, Node.js, .NET, Python, Go, or Ruby agents, you can [add an application exclusion \(page 1142\)](#) under policy management, or from the list of attack events.

To view a list of existing exclusions, go to **Applications > Your application name > Policy > Exclusions** or **user menu > Policy management > Application exclusions**.

To add application exclusions for a specific application, go to **Applications > Your application name > Policy > Exclusions** or **Attacks > Attack Events**.

## Add application exclusions

Java, .NET Framework, .NET Core, Node.js, Python, Go, and Ruby agents let you use an [application exclusion \(page 1142\)](#) to exclude certain applications, or parts of them, from security analysis.

Currently, PHP agents do not support application exclusions.

## Before you begin

- Access control requirement:
  - **Users and groups:** Organization RulesAdmin or Organization Admin role.
  - **Role-based access control (preview feature):** Role that includes the Edit application action.
- Java and .NET agents support code exclusions.
- Java, .NET, Node.js, Python, Ruby, and Go agents support input exclusions.
- Java, .NET, Node.js, Python, and Ruby agents support URL exclusions.
- Java and .NET Core agents support queue/topic (message queue) exclusions.

## Steps

1. Select **Applications** in the header and select the name of your application. Exclusions only apply to the application for which they were created.
2. Select the **Policy** tab and then, select **Exclusions**.
3. Select **Add Exclusion**.



### TIP

You can also create an exclusion from an existing attack event. When viewing the list of attack events, **Attacks > Attack events**, select the triangle in the far right column, then select **Add exclusion**. Selecting this button pre-populates the exclusion fields based on the details of this specific event.

Once created, this exclusion is visible in the list of exclusions.

4. In Add Exclusions, enter a **Name** for this exclusion (something you'll remember easily).

## 5. Select the **Exclusion type**.

Input, URL and Queue/topic-based exclusion definitions accept a subset of Perl Compatible Regular Expressions (PCRE) which includes these values:

```
. * for 0 or more of any character
.+ for 1 or more of any character
.? for 0 or 1 of any character
. for 1 of any character
\ . for an escaped literal of . for usage Example: somefile\.jsp
```

Use these [examples \(page 1147\)](#) to guide you.

Select one of these options:

- **Code:** Enter the method signatures you want to be suppressed. For example, if you have a method called `doLegacySecurity()` inside a class called `com.Acme.OldSecurity` that is being reported for using insecure cryptographic algorithms, you can ignore it by entering:

```
Com.Acme.OldSecurity.DoLegacySecurity
```

Be sure to include the entire method signature without a trailing parameter definition or any other extra characters. Contrast matches this method signature against the stack trace for any vulnerabilities found. Contrast suppresses any method signatures containing a match.

- **Input:** Enter an input type and an input name. Any findings using this input will be suppressed. [Input exclusions \(page 1144\)](#) provides details and examples on using these exclusions.
  - For **Parameter**, **Header** and **Cookie**: You must specify the name of the particular input for which you wish to suppress findings. You can use wildcard `*` to suppress all findings from the selected input type.
  - **QueryString** and **Body**: These will suppress findings from the entire QueryString and Body, respectively. The QueryString and Body may only be excluded in conjunction with the URL exclusion pattern defined below.

For the **Input** exclusion type, under Applied URLs, choose how to apply URLs:

- **All URLs:** Findings using the specified input type and name will be suppressed regardless of their origin.
- **These URLs:** Specify a set of paths to which to apply the exclusion. You can use [regex \(page 1123\)](#) and [wildcard expressions. \(page 1147\)](#)



### IMPORTANT

Do not include protocol schemes (`http://` or `https://`) or hostnames; only use path names beginning with `/`.

Slash followed by dot-wildcard `/ . *` is an acceptable substitute for listing all URLs.

Designate URLs that should be ignored by certain rules.

- **URL:** Designate URLs that should be ignored by certain rules. List the URL paths to be excluded, one per line. You can use [regex \(page 1123\)](#) and [wildcard expressions \(page 1147\)](#).
- **Queue/topic:** Specify a message queue or topic that should be ignored by certain rules. A message queue has one consumer while a topic has multiple consumers. Currently, this option is supported by Java only.

For the Queue/topic exclusion type, under Applied queues, choose how to apply the queues or topic names:

- **All queues/topics:** Findings from all queues and topics are suppressed.
- **These queues:** Specify a list of queue or topic names to be excluded. You can specify queue names or use [regex. \(page 1123\)](#) and [wildcard expressions \(page 1147\)](#).

- Under **Applicable rules**, specify the scope of rules affected by the exclusion. All rules is the default, or you can click in the box to select multiple options:

- **All rules** applies the exclusion to all vulnerabilities found in both Assess and Protect mode.
  - Under Assess, **All Assess rules** applies to all vulnerabilities found when Assess is enabled.
  - Under Protect, **All Protect rules** applies to all attack events when Protect is enabled.
  - Under the Assess section or the Protect section, selecting individual rules lets you further narrow the focus. Exclusions are only applied to vulnerabilities that the selected rules find.  
If you select **Input** as the exclusion type, you can only select rules that are not triggered by user input.
  - Under Assess and Protect, select individual rules found in both Assess and Protect mode.
7. Select the box next to **Suppress all events that match this exclusion** if you want Contrast to suppress historical events that have already been reported.
  8. Select **Add**.  
The exclusion is added to the list of exclusions. Any inputs that match the criteria you entered won't be processed with the rules you've applied.  
You can view this list either at **Applications > Your application name > Policy > Exclusions** or in the **user menu > Policy management > Application exclusions**. From the list, you can use the toggles to enable or disable the exclusion for Assess or Protect.

## Input exclusions

One type of an application exclusion is an input exclusion. These exclusions let you exclude specific inputs from a Contrast agent's analysis. This exclusion type can be useful if you know certain inputs are safe and do not require monitoring, thereby reducing noise in your security reports.

## Types of input exclusions

Contrast supports exclusions that use these types of inputs:

- **Parameter:** Excludes specific parameters accessed through request methods, including parameters from the post body or query string itself.
- **Header:** Excludes specific HTTP headers.
- **Query String:** Excludes the entire query string.
- **Body:** Excludes the entire request body.
- **Cookie:** Excludes specific cookies.

## Parameter Input exclusions

An input exclusion with an input type of parameter checks for any parameters in the request, like those from the query string or form body, that match the specified parameter, as long as the application retrieves it using that parameter name (for example, `request.getParameter("foo")`).

### Example:

Add exclusion (\*\*Terracotta Bank)
×

Use an application exclusion to suppress events when you use security controls outside of Contrast agent's scope. [Learn more](#)

Exclusion name

Parameter exclusiong

Exclusion type

Input

Input type


Parameter

Input name ?

foo

If you specify `foo` as the parameter to exclude, the following table shows examples of how Contrast handles the HTTP request.

HTTP request	If excluded...	If included...
GET /someRequest?  foo=excludedValue&bar=includedValue	Contrast excludes this parameter from monitoring:  foo=excludedValue	Contrast continues to monitor this parameter:  bar=includedValue
POST /someRequest  Content-Type: application/x-www-form-urlencoded  foo=excludedValue&bar=includedValue	Contrast excludes this parameter from monitoring:  foo=excludedValue	Contrast continues to monitor this parameter:  bar=includedValue



**NOTE**  
For a POST request, the exclusion only works for a body format of `application/x-www-form-urlencoded` body.

## Header input exclusions

Headers are specific HTTP headers that the application retrieves from the request (for example, `request.getHeader("User-Agent")`).

### Example:

Add exclusion (\*\*Terracotta Bank)
×

Use an application exclusion to suppress events when you use security controls outside of Contrast agent's scope. [Learn more](#)

Exclusion name

Exclusion type

Input type

Input name ?

If you specify `User-Agent` as a header to exclude, the following table shows examples of how Contrast handles HTTP requests.

HTTP request	If excluded...	If included...
GET /someRequest  User-Agent: excludedUser  AgentAccept: application/json	Contrast excludes this header from monitoring:  User-Agent: excludedUserAgent	Contrast continues to monitor this header:  Accept: application/json

## Query string input exclusions

The query string exclusion applies to the entire query string, not individual parameters. This exclusion is useful when the application retrieves and parses the entire query string (for example, acting on the result of `request.getQueryString()`).

### Example:

**Add exclusion** (\*\*Terracotta Bank) ×  
Use an application exclusion to suppress events when you use security controls outside of Contrast agent's scope. [Learn more](#)

Exclusion name

Query exclusion

Exclusion type

Input

Input type

Query String

Applied URLs

☒ All URLs

☐ These URLs

Based on specified URLs, the following table shows examples of how Contrast handles HTTP requests.

HTTP request	If excluded...	If included...
GET /someRequest? foo=excludedValue&bar=excludedValue	Contrast excludes the entire query string from monitoring	Contrast continues to monitor the entire query string

## Body input exclusions

The body input exclusion applies to the entire body of the request. This type of exclusion is useful when the body contains data that the application parses (for example, acting on the result of `request.getBody()`)

### Example:

**Add exclusion** (\*\*Terracotta Bank) ×  
Use an application exclusion to suppress events when you use security controls outside of Contrast agent's scope. [Learn more](#)

Exclusion name

Body exclusion

Exclusion type

Input

Input type

Body

Applied URLs

☒ All URLs

☐ These URLs

Based on specified URLs, the following table shows examples of how Contrast handles HTTP requests.

HTTP request	If excluded...	If included...
POST /someRequest  Content-Type: application/json  { "foo": "excludedValue", "bar": "excludedValue" }	Contrast excludes the entire request body from monitoring	Contrast continues to monitor the entire request body

## Cookie input exclusions

Cookie input exclusions are useful when you want to suppress findings related to specified cookies.

### Example:

Add exclusion (\*\*Terracotta Bank)
×

Use an application exclusion to suppress events when you use security controls outside of Contrast agent's scope. [Learn more](#)

Exclusion name

Exclusion type

Input type

Input name ?

If you specify `sessionID` as the cookie to exclude, the following table shows examples of how Contrast handles HTTP requests.

HTTP request	If excluded...	If included...
GET /someRequest	Contrast excludes this cookie from monitoring:	Contrast continues to monitor this cookie:
Cookie: sessionID=excludedValue; otherCookie=includedValue	sessionID=excludedValue	otherCookie=includedValue

## Wildcard expressions

When specifying Input, URL, or Queue/topic exclusions for applications, you can build wildcard expressions using:

- `.*` to mean 0 or more of any character
  - `.+` to mean 1 or more of any character
  - `.?` to mean 0 or 1 of any character
  - `.` to mean 1 of any character
  - `\.` for an escaped literal of `.` for usage
- Example: `somefile\.jsp`

## Wildcard expression examples

Desired effect	Regular expression	Example
Exclude all subpaths	<code>/myapp/.+</code>	Excludes all paths with the initial URL of <code>/myapp/</code>
Exclude one character from subpath	<code>/.yapp</code>	Excludes all subpaths that are 5 characters and end in <code>yapp</code> (like <code>myapp</code> )
Exclude one subpath explicitly	<code>/myapp/thispath</code>	Excludes only <code>/myapp/thispath</code>
Exclude path ending	<code>/*.ignore</code>	Excludes all paths ending in <code>ignore</code>
Exclude paths containing	<code>/*.value.*</code>	Excludes all paths containing <code>value</code>
Exclude path containing	<code>/.?value.*</code>	Excludes all paths either starting with <code>value</code> or paths that have one character before <code>value</code>
Exclude paths where a period (dot) is used as a literal character and not a wildcard.	<code>/myapp\.js</code>	Excludes only <code>myapp.js</code>  You can use up to three instances of this expression.

## Set compliance policy

You can define compliance policies for application compliance within your organization. If any designated applications violate this policy, Contrast marks them so you can quickly find them and fix them. (Administrators are also notified of violations by email.)

To set compliance policy:

1. Under [policy management \(page 1118\)](#), select **Compliance policy**.
2. You will see a list of existing compliance policies if there are any. You can enable or disable policies using the toggles, or delete them with the **Delete** icon.

3. Select the name of any policy to edit, or select **Add policy** at the top of the grid to create a new compliance policy.
4. In the panel that opens enter:
  - **Name:** Choose a name for the policy.
  - **Policy criteria:** The default is **All rules**, or you can type ahead and select vulnerabilities by severity level(s), security standards or Assess rules.
  - **Applications:** The default is All applications or you can type ahead and select applications by level(s) of importance and/or individual name.
5. Select **Add** or **Save**.

**NOTE**

For default policies, the **Name** and **Policy criteria** fields are locked, and you cannot delete them. However, you can modify application selections for default policies.

**TIP**

Enabled policies can be used to filter applications by compliance policy. To do this select Applications. In the Applications page, click the Advanced link to filter application by Compliance Policy.

**NOTE**

If an applicable vulnerability isn't remediated, or applicable Security Standards and Assess Rules are being violated, Contrast flags the corresponding applications in the Applications page. Hover over the warning icon in the Applications grid or go to the application's details page for a link to the violated policy.

## IP management

Manage IP policy in your organization with denylists, allowlists (trusted hosts), and source names:

**NOTE**

For denylists and allowlists, Contrast checks the `Client Address` and the `X-Forwarded-For` request headers to see if the IP addresses match the list entries.

- **IP denylist:** Sets rules that let Contrast Protect block all IP addresses in this list  
Using a denylist is appropriate for immediate triage until you can put a more permanent Protect policy in place or conduct an investigation.
- **IP allowlist:** Marks trusted hosts conducting internal vulnerability scans as safe. Contrast doesn't show data for IP addresses in this list.



Entries in this list don't override entries in IP denylists

Contrast Assess features remain unaffected and continue to function as normal.

Contrast Protect ignores all IP addresses (or ranges) that match entries in this list. It does not monitor or block any attacks from IP addresses in the list.

- **Source name:** Labels attack events caused by known sources, such as pen testers, based on one or more IP addresses or subnet masks.

When you view attacks in the **Attacks > Monitor** and **Attack Details** pages, Contrast displays the source name instead of the attacker's IP information. Displaying this value allows you to quickly identify and differentiate expected events from attack events that need your attention.

## See also

- [Block or allow IPs \(page 1149\)](#)
- [Manage source names \(page 1150\)](#)

## Deny or allow IP addresses

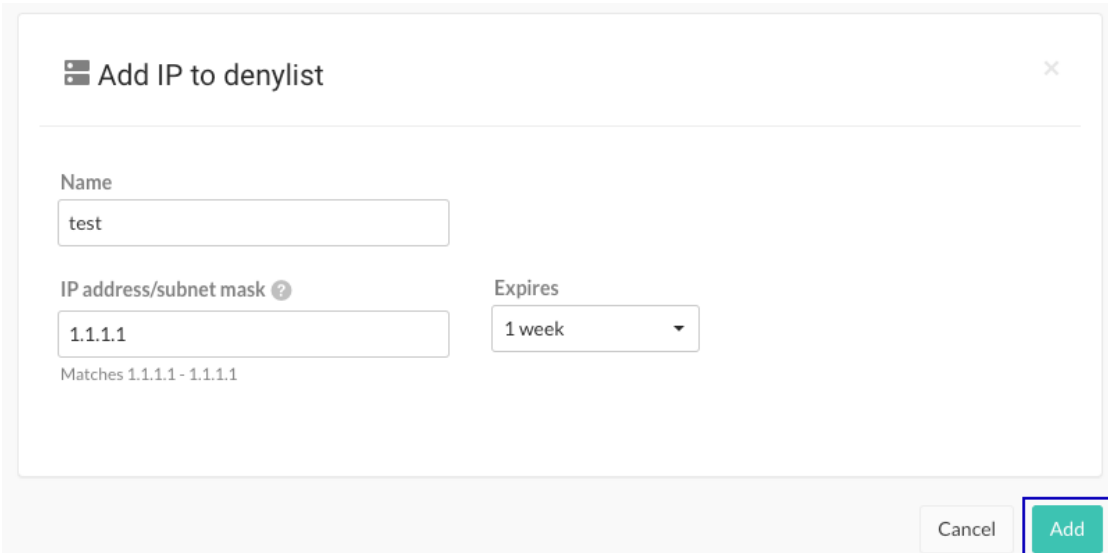
Use IP denylists or IP allowlists to [manage IP addresses \(page 1148\)](#) in your organization.

## Before you begin

- You must have the organization role of Admin or RulesAdmin role.
- You can use Classless Inter Domain Routing (CIDR) notation to specify a subnet mask.

## Steps

1. From the user menu, select **Policy Management > IP Management**.
2. Manage denylists:
  - a. Select the **IP denylist** tab.
  - b. To edit a denylist, select a denylist name, change the displayed information, and select **Save**.
  - c. To add an IP address to a denylist, select **Add IP to denylist**, specify the details, and select **Add**.



**Add IP to denylist**

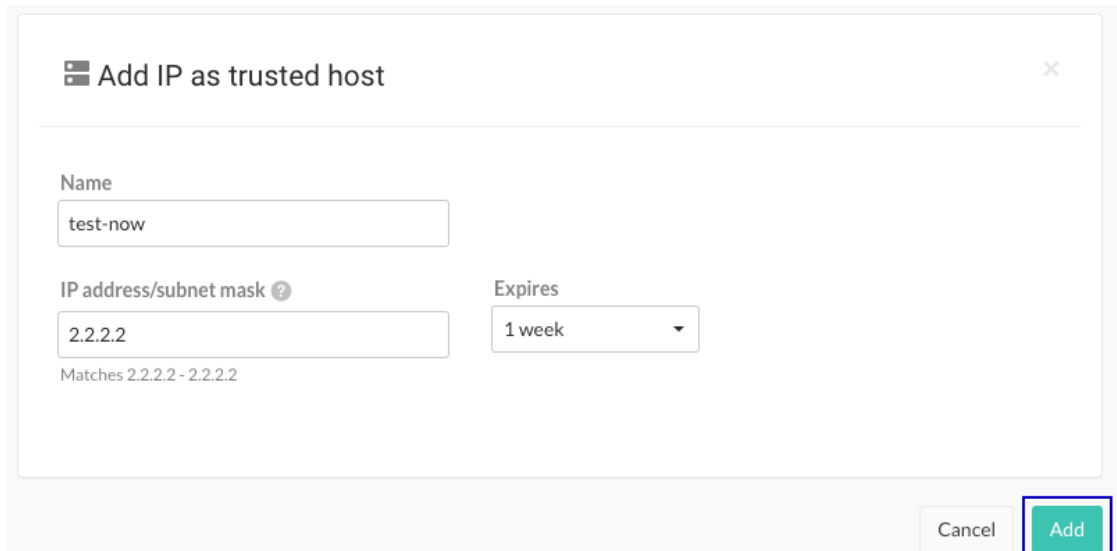
Name  
test

IP address/subnet mask ?  
1.1.1.1  
Matches 1.1.1.1 - 1.1.1.1

Expires  
1 week

Cancel Add

3. Manage allowlists:
  - a. Select the **IP allowlist** tab.
  - b. To edit an allowlist, select the allowlist name, change the displayed information, and select **Save**.
  - c. To add an trusted host to an allowlist, select **Add trusted host**, specify the details, and select **Add**.



## Manage source names

Use source names to quickly identify non-threatening, internal traffic and testing, while monitoring attack events in your organization.

You can label one or more IP addresses and subnet masks with a source name of your choice. When the source name is saved, you can see the name (rather than user IP information) by selecting **Attacks > Monitor** or looking on the **Attacks details** page. This can make it easier to identify the named attacker as a known source when assessing attack events.

To create source names:

1. Go to the **user menu > Policy management > IP management > Source names**.
2. Select **Add source name**.
3. Enter the **Name** you want to use to identify one or more IP addresses.
4. Add the **IP address/Subnet mask** to identify with this source name. Use the link to **Add** more IP addresses or subnet masks to the group, if necessary.
5. Use the dropdowns to select the **Start** and **End** dates and times for the source name. You may choose to create a custom time span that starts on a past date; in this case, the source name applies retroactively to any attack events.
6. Once the fields are completed, select **Add** to save the source name.  
Once a source name is added in your organization, the source name appears for attacks that match the criteria on the **Monitor** and **Attack** details pages. This will help you [monitor attacks \(page 588\)](#).  
If the data reported for an attack event matches more than one source name, Contrast applies the name that you updated most recently.
7. To edit a source name later, select the source name. In the **Edit source name** form that opens, make changes and select **Save**.
8. To delete a source name, either select the **Delete** icon in the **Source names grid**, or below the **Edit source names** form. Once the name is deleted, all references to the name are replaced with the IP information.

## Set library policy



### IMPORTANT

License policy is available to Contrast SCA customers only. Contact your Organization Administrator to enable SCA.

Contrast can flag libraries that don't meet your organization's criteria to ensure your applications are secure.

If a library is restricted or used in an application that's below a specific version, it's marked as a policy violation by Contrast. You can also tell Contrast to automatically grade any library that violates the policy with the letter "F" to flag it in the Contrast interface. (Administrators are notified of violations in both the product and by email.)



### NOTE

Library versions include any major, minor, and patch versions.

To set a library policy:

1. In the user menu, select **Policy Management > Library Policy**.
2. Check the box to **Restrict libraries** and choose which libraries you want to exclude from your portfolio. You can select multiple.
3. Check the box to **Enable version requirements** and choose one or multiple libraries that must be within your given number of versions.
4. Click the **Add another requirement** link to create version requirements for additional library groupings.
5. Check the **Restrict licenses** box to set a policy on open-source licenses that you want to restrict. If an open-source license is restricted, then any libraries that use the restricted license will be marked as a policy violation.  
The license policy lists open-source licenses in SPDX format, listed by short identifier and followed by the full name. Any license type that you want to restrict must be selected. Contrast includes any 'or later' licenses it identifies in your portfolio. For example, if you restrict by GPL-3.0-only, any licenses that are GPL-3.0-or-later will be included in that restriction.
6. Check the box next to **Fail libraries in violation of policy**, to automatically assign a failing score to any library that violates a set policy.  
If a library fails to comply with a set policy, the name, a warning icon and the library score are highlighted in red in the **Libraries** page. Hover over the icon or go to the library's **Overview** page for more information about the violation.  
If you choose to automatically fail libraries, Organization Administrators will be notified when [adjusting score settings \(page 1177\)](#).

## Sensitive data masking

Sensitive data masking limits risk to your organization and helps meet compliance requirements.

Data masking protects sensitive data in your applications by redacting it in vulnerability and attack reports that are sent to Contrast, syslog or security log.

Contrast offers several categories of sensitive data, or data types, that are comprised of specific keywords that the agent automatically identifies and redacts in reports. A user with at least RulesAdmin permissions can [manage sensitive data \(page 1152\)](#).

Contrast agents mask sensitive data in query parameters, request headers, cookies and body. Your agent identifies sensitive data by searching for specific keywords used in the input name. If the agent finds a match, it redacts the value for that input, and replaces it with a placeholder with the format `contrast-redacted-{datatype}`, where `datatype` is the category of sensitive data to which the keyword belongs.

Contrast agents do **not** mask individual fields in request bodies with a content type other than `application/x-www-form-urlencoded`; however, you can configure the agent to mask the entire request body. Contrast agents also do not mask data that appears in the data flow portion of a vulnerability report, if using Assess, or in the vector of an attack event, if using Protect.



## NOTE

Contrast agents make a “best effort” attempt to avoid printing sensitive data in Contrast log statements; however, it’s possible that sensitive data could appear in the Contrast log, if the log level is set to DEBUG or lower. Whenever possible, you should avoid setting production systems to log at DEBUG or lower. If a system that deals with sensitive data is set to log at DEBUG or lower, you should take steps to ensure that those logs are not being sent to an external system to avoid leaking any sensitive data.

For example, this HTTP request sent by an agent as part of a vulnerability report shows two inputs that the agent identified as sensitive, as well as the placeholders it used to mask the values of the input before sending the report to Contrast, syslog server or security log.

```
PUT /employee/5 HTTP/1.1
Host: yourdomain.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 30
apikey: contrast-redacted-authentication-info

ssn=contrast-redacted-government-id&department=sales
```

In this case, the header value is masked because “apikey” matches a keyword in the “Authentication Info” data type, and the form parameter is redacted because “ssn” matches a keyword in the “Government ID” data type for your Contrast organization. (Keyword matches are case insensitive.)

## Manage sensitive data types

[Sensitive data masking \(page 1151\)](#) limits risk to your organization and helps meet compliance requirements.

1. Under policy management, select **Sensitive data**.
2. Here you can see an alphabetical list of sensitive data types. Use the search field to find a particular type by name or keyword.
3. Select the checkbox next to **Mask entire body** to enable redaction of the entire HTTP request body. This setting applies to all applications in your organization.
4. To mask sensitive data in attack values, select the checkbox next to **Mask sensitive data in attack values**.

Masking sensitive data in attack values can decrease your ability to evaluate the impact of an attack event. Contrast agents do not normally mask this type of data

5. Critical data types and keywords determined by Contrast apply to all applications in your organization by default, and can't be edited or disabled. For data types that Contrast has not determined to be critical, you may use the toggle in the grid to enable or disable them for the organization.
6. To add custom keywords, select the name of the data type in the list. In the Custom keywords box at the bottom of the page, enter one or more custom keywords and select **Add**. Custom keywords apply to all applications in your organization.
7. Select **Save**.

## Add and edit notifications as a RulesAdmin

Notification defaults are set by an Organization Administrator, but as a RulesAdmin you can enable or disable existing notifications or create new ones.

1. Under [organization settings \(page 1158\)](#), select **Notifications**.
2. Use the toggles to enable or disable existing notifications.
3. To create a new notification, select **Create notification**.
4. In the window that appears, enter:
  - Name
  - Frequency
  - Description
  - Applications
  - Application tags
  - Users
5. Select **Save**.

## Organization administration

If you have Organization Administrator permissions you can do everything a [RulesAdmin \(page 1118\)](#) or [Editor \(page 593\)](#) can do, plus, at an organization level you can:

- [Configure organization settings \(page 1158\)](#)
- [Enable Assess \(page 1153\)](#)
- [Enable Protect \(page 1155\)](#)



### NOTE

Organization Administrators have the highest organization level permissions. Other [organization roles \(page 1267\)](#) also have capabilities that span across their organization.

If [configured at a system level \(page 1224\)](#), users can fill the Organization Administrator role across multiple organizations.

## Enable Assess

You can use configuration files, variables, or the Contrast web interface to configure the Assess setting. This procedure describes how to enable Assess in the Contrast web interface. [YAML configuration \(page 107\)](#) and [Environment variables \(page 110\)](#) describe how to use methods outside of the web interface to configure agent settings.

## Before you begin

- Although you can see the types of vulnerabilities that Contrast discovers without an Assess license, you won't be able to retrieve any details unless you apply licenses to applications.

## Steps

1. Log in to Contrast.
2. Select **Servers** in the header.
3. Either scroll or use the search at the top of the page to find the servers associated with the applications you want to analyze with Assess.
4. To manage the Assess setting for specific servers in the Contrast web interface, use either of these methods:
  - In the Servers list, select the setting in the Assess column.
  - In the Servers list, select a server name to open its detailed view and use the Assess setting there.

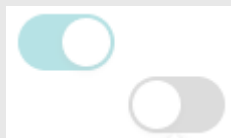


### NOTE

- If you use only the Contrast web interface to turn Assess on or off, the Assess setting is green if ON and gray if OFF. You can change this setting in the Contrast web interface.

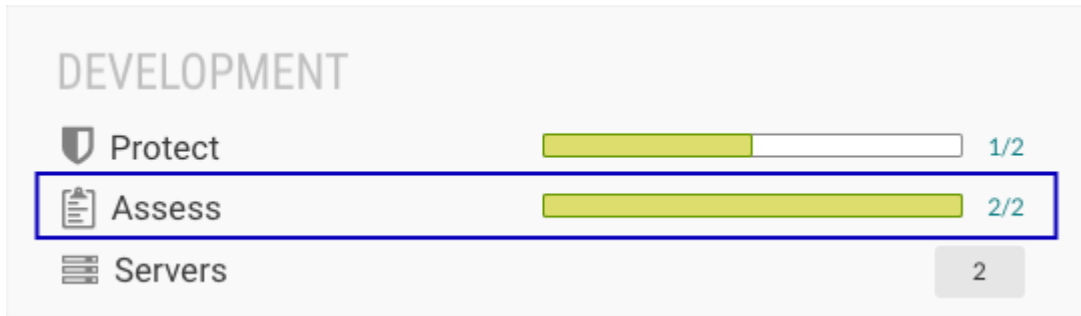


- If you used a method external to the Contrast web interface to configure the setting for Assess (for example, an agent configuration file), the setting is green but disabled if ON and grey but disabled if OFF. You cannot change this setting in the Contrast web interface.




- If the setting in the Contrast web interface is disabled, hover over the setting to see where it is configured. The [order of precedence \(page 106\)](#) determines which setting Contrast uses as the effective configuration.













5. To determine if an application is using Assess on each server associated with it, go to the Applications page:
  - a. Select **Applications** in the header.
  - b. Select an application.
  - c. In the Overview tab, under each environment, if at least one server has Assess turned on, the bar next to the Assess setting indicates the Assess status for all servers associated with the application. A green bar represents the number of servers that have Assess turned on. A white bar represents the number of servers that have Assess turned off.



If no servers have Assess turned on, you see an OFF icon (  ).

- d. To see if the application is configured to use Assess for each server associated with it, select a section of the Assess setting bar to open a filtered view of the Servers list.

**Servers** All (4751)  ↓ SORT BY LAST SEEN

Server ▼	Last Seen	Environment ▼	Applications	Assess	Protect
ip-172-5.0.0.216 	4 hours ago	Development	PF:1681275975		
ip-172-5.0.0.216 	4 hours ago	Development	PF:1681275940		
ip-172-5.0.0 	9 hours ago	Development	PF:1681269945		
UMAHESWARI-C02DD09CMD6R 4.13.1 	20 hours ago	Development	application		

6. To set a default Assess setting for new servers, from the user menu, select **Organization settings > Servers** and use the Assess setting there.
7. If the application you want to analyze with Assess is unlicensed, add a license to it:
  - a. Select **Applications** in the header.
  - b. Select **Unlicensed** next to the application name
  - c. In the Apply License window, select **Apply license**.
  - d. Restart the application server to ensure the Contrast agent instruments your application with Assess capabilities. Once that's complete, Contrast begins to receive vulnerability analytics. The application no longer has **Unlicensed** next to it, which means there is an Assess license assigned to it.



## NOTE

Organization Administrators can configure a default setting that applies Assess licenses automatically to new applications rather than applying licenses manually for every application.

1. Under [organization settings \(page 1158\)](#), select **Organization**.
2. In Licenses, under Assess, select **Automatically apply licenses to new applications**.

## Enable Protect

Enabling Protect for users lets them access and see Protect data. Enabling Protect for servers lets applications use Protect to monitor and block attacks.


**NOTE**

If you enable Protect on servers with existing applications, restart the applications to have Protect take effect.

## Before you begin

- Go to **Organization settings > Users** and verify that you have permissions to access Protect data and settings.
  - For hosted customers, Contrast grants Protect permissions to organizations and user roles in the organization.
  - For on-premises customers, SuperAdmin, ServerAdmin, or System Administrator roles are required to [grant Protect permissions for one or more organizations. \(page 1226\)](#)  
These roles can also configure which user roles have access to Protect data.
- Ensure that you have [licenses \(page 1161\)](#) that you can apply to servers.
- To enable Protect for users, an Organization Administrator role is required.

## Let users access Protect data

1. Log in to Contrast.
2. Enable users to see and use Protect data:
  - a. In the user menu, select **Organization Settings**.
  - b. Select **Users**.
  - c. For each user who needs access to Protect data, turn on the Protect setting (  )
  - d. To have the new setting take effect, tell users to log out of the Contrast web interface and log in again.

## Enable Protect for servers

You can use configuration files, variables, or the Contrast web interface to configure the Protect setting. This procedure describes how to enable Protect in the Contrast web interface. [YAML configuration \(page 107\)](#) and [Environment variables \(page 110\)](#) describe how to use methods outside of the web interface to configure agent settings.

1. To configure a default Protect setting for new servers in selected environments:

If you use a method outside the Contrast web interface to configure the Protect setting for a specific server, that configuration overrides this default setting.

  - a. From the user menu, select **Organization settings**.
  - b. Select **Servers (page 1171)**.
  - c. Select an environment.
  - d. Under Protect, turn on the Protect setting.




Protect

**NOTE**

To automatically [allocate licenses \(page 1161\)](#), an Organization Administrator or RulesAdmins role is required. This option is useful if you don't want to enable Protect for servers, one at a time.



2. Enable Protect for specific servers:
  - a. Select **Servers** in the header.
  - b. To enable Protect in the Contrast web interface, turn on the Protect setting (  ). Use either of these methods:
    - In the Servers list, turn on the setting in the **Protect** column.
    - In the Servers list, select a server name and in the Overview tab, turn on the Protect setting.



#### NOTE

- If you use only the Contrast web interface to turn Protect on or off, the Protect setting for a specific server is green if ON and gray if OFF. You can change this setting in the Contrast web interface.



- If you used a method external to the Contrast web interface to configure the setting for Protect (for example, an agent configuration file), the setting is green but disabled if ON and gray but disabled if OFF. You cannot change this setting in the Contrast web interface.

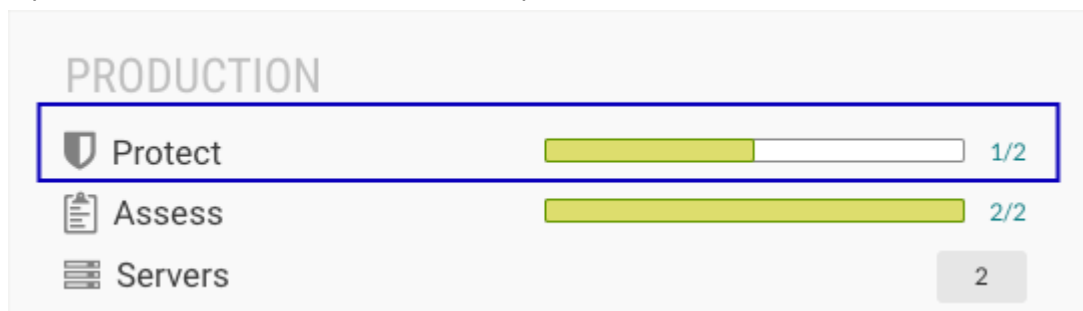


- If the setting in the Contrast web interface is disabled, hover over the setting to see where it is configured. The [order of precedence \(page 106\)](#) determines which setting Contrast uses as the effective configuration.

3. To verify that Protect is turned on for a specific server, in the Servers tab, select the server, select **Overview** and verify the Protect setting is green.
  - If one or more applications associated with the server are not configured to use Protect, a warning icon displays next to the Protect setting.

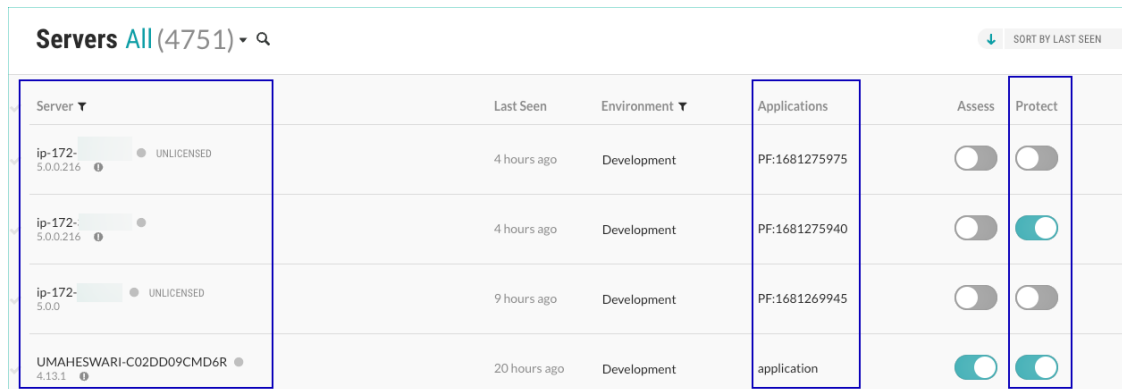










4. To determine if an application is using Protect on each server associated with it, go to the Applications page:
  - a. Select **Applications** in the header.
  - b. Select an application.
  - c. In the Overview tab, under each environment, if at least one server has Protect turned on, the bar next to the Protect setting indicates the Protect status for all servers associated with the application. A green bar represents the number of servers that are protected. A white bar represents the number servers that are not protected.



If no servers have Protect turned on, you see an Off icon (  ).

- d. To see if the application is configured to use Protect for each server associated with it, select a section of the Protect setting bar to open a filtered view of the Servers list.



Server ▼	Last Seen	Environment ▼	Applications	Assess	Protect
ip-172-5.0.0.216 ● UNLICENSED	4 hours ago	Development	PF:1681275975		
ip-172-5.0.0.216 ●	4 hours ago	Development	PF:1681275940		
ip-172-5.0.0.0 ● UNLICENSED	9 hours ago	Development	PF:1681269945		
UMAHESWARI-C02DD09CMD6R ● 4.13.1	20 hours ago	Development	application		

## License behavior

Contrast applies Protect licenses automatically to servers when these conditions exist:

- Protect is turned on for an organization.
- Automatic application of Protect licenses is turned on for an organization and Protect is NOT enabled by a method external to the Contrast web interface (for example, an agent configuration file).  
If Protect is enabled with a method external to the Contrast web interface, that effective configuration overrides the automatic licensing option in the Contrast web interface.
- A server exists in one or more environments where automatic licensing is turned on.

If you also use the Protect setting in an agent configuration file, it overrides the license behavior in the following ways:

- **Protect is turned on in the agent configuration file**
  - If Protect licenses are available when the application starts, you might notice the server is licensed for a very brief period of time. Contrast removes the license automatically as soon as the agent registers the application.
  - If **no** Protect licenses are available when the application starts, Contrast tries to apply a license to the server every time the agent communicates with Contrast.
- **Protect is turned off in the agent configuration file**
  - If Protect licenses are available when the application starts, Contrast tries to apply a license to the server. When an agent registers an application with Contrast, it removes the license applied to the server.
  - If **no** Protect licenses are available when the application starts, Contrast doesn't apply a license to the server.

## Configure organization settings

If you have Organization Administrator permissions you can configure settings for your organizations:

## Steps

1. If you have multiple organizations, select the name of the organization you want to configure in the **user menu**.
2. Select **Organization settings** in the **user menu**.  
The organization settings are:
  - Organization ([general information \(page 1160\)](#), [licenses \(page 1161\)](#), and [usage analytics \(page 1167\)](#))
  - [Users, groups and permissions \(page 1163\)](#) including [API only users \(page 1165\)](#).
  - Security ([passwords \(page 1168\)](#), [multi-factor authentication \(page 1168\)](#), [IP range \(page 1168\)](#), [email domain restrictions \(page 1168\)](#))
  - Agent (view the [agent keys \(page 104\)](#) that you use to configure agents to communicate with Contrast)
  - [Single sign-on \(SSO\) \(page 1169\)](#)

If you're not using SSO, protect your system by configuring [multi-factor authentication \(page 1168\)](#) as required. Contrast supports configurations that use both SSO and multi-factor authentication.

- [Integrations \(page 1051\)](#)
- [Servers \(page 1171\)](#)
- [Applications \(page 1173\)](#)
- [Notifications \(page 1175\)](#)
- [Score Settings \(page 1177\)](#)

## Configure general organization information

1. Under [organization settings \(page 1158\)](#), select **Organization** in the left navigation.
2. In the top right corner you can see the UUID for this organization. (This is helpful in [bulk adding users \(page 1222\)](#).)
3. This panel also shows general information about the organization like:
  - Organization name
  - Default time and date formats for the organization
  - Default language settings for the organization such as Japanese (if enabled by a superadmin), or English



### NOTE

Individual user settings override most general organization settings (time and date formats, for example) with the exception of language settings. User language settings override organization language settings for things specific to the user, such as user notifications or emails. However, user language settings will not override organization language settings for things specific to the organization, such as organization-level notifications.

4. Select **Edit** to change any of the information on this panel.
5. Make changes and select **Save**.

## Manage AI settings at an organization level

An administrator can control whether AI guidance is available for providing additional information and suggestions for vulnerabilities.



### REMINDER

Use of Intelligent Remediation Guidance (the "Feature") means that you agree to submit data to Anthropic on AWS Bedrock that will be analyzed and used as authorized under the [Anthropic on Bedrock Terms of Service](#). Use of the Feature is entirely at your own risk.

## Before you begin

- If you are using role-based access control, you need a role with the Manage organization action.
- If you are using organization groups and users, you need the Organization Admin role.

## Steps

1. Under organization settings, select **General**.
2. Under AI settings, select **Intelligent Remediation guidance** to turn this feature on or off.
3. Optional: Select **Show legal disclaimer for everyone** if you want users to see the legal disclaimer when they use Contrast AI guidance.

**See also:**

[Contrast AI guidance \(page 1022\)](#)

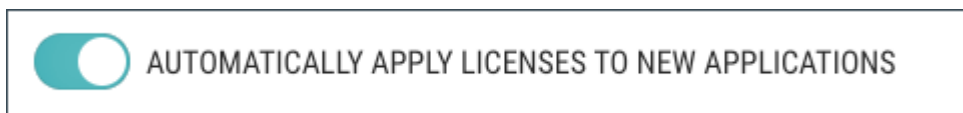
## Allocate licenses for organizations, applications, and servers

### Before you begin

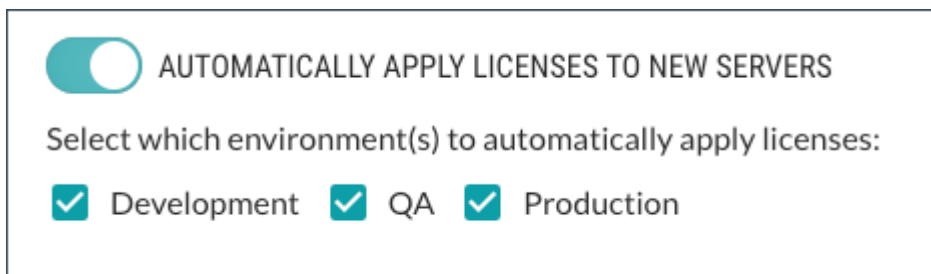
- An Organization Administrator role is required.
- If your organization has consumed all allocated Assess licenses, hosted customers are asked to [contact Support](#).
- For on-premises customers, the SuperAdmin can [make more licenses available at a system level \(page 1251\)](#).

### Steps

1. See an overview of license usage in an organization:
  - a. From the user menu, select **Organization settings**.
  - b. Select **Organization**.
  - c. Under **Licensing**, view information about Assess and Protect licenses:
    - Available and used Assess (application) licenses, as well as how many applications are unlicensed.
    - Available and used Protect (server) licenses as well as how many servers are unlicensed. If you are using more licenses than you purchased, the available licenses section of the license bar is replaced with the number of additional licenses in use.
2. (Optional) Under **Licensing**, automatically apply licenses to new applications or servers:
  - a. For Assess licenses, turn on **Automatically apply licenses to new applications**.



- b. For Protect licenses, turn on **Automatically apply licenses to new servers**. Select the server environments.



If you use a method external to the Contrast web interface for Protect settings (for example, an agent configuration file), Contrast considers that configuration to be the effective configuration. An effective configuration takes precedence over the automatic licensing setting in the Contrast web interface:

- If an agent's effective configuration has Protect turned OFF and the automatic licensing setting is turned ON, Contrast ignores the setting in the Contrast web interface. It doesn't apply a license to the server automatically.

- If an agent's effective configuration has Protect turned ON and the automatic licensing setting is turned OFF, Contrast ignores the setting in the Contrast web interface. It applies a license to the server automatically.
  - If an agent doesn't report an effective configuration, the automatic licensing setting in the Contrast web interface applies to all new servers.
3. If needed, apply Assess licenses to individual applications:
    - a. Select **Applications** in the header.  
In the applications list, applications that are unlicensed show **Unlicensed** after their name. You cannot view vulnerability data for the application if it is unlicensed.
    - b. To apply a license to an unlicensed application, select **Unlicensed**.
    - c. In the Apply License window, select **Apply License**.
  4. If needed, remove Assess licenses from applications by [deleting the application \(page 616\)](#).  
Licenses applied to applications permanently count towards the number of maximum allowable applications. Deleting a licensed application has no effect on the number of licenses you are allowed to apply to applications
  5. If needed, add or remove Protect licenses from individual servers:
    - a. Select **Servers** in the header.
    - b. Turn the Protect setting on or off for the server.
      - If you use only the Contrast web interface to turn Protect on or off, the Protect setting for a specific server is green if ON and gray if OFF. You can change this setting in the Contrast web interface.



- If you used a method external to the Contrast web interface to configure the setting for Protect (for example, an agent configuration file), the setting is green but disabled if ON and gray but disabled if OFF. You cannot change this setting in the Contrast web interface.



- If the setting in the Contrast web interface is disabled, hover over the setting to see where it is configured. The [order of precedence \(page 106\)](#) determines which setting Contrast uses as the effective configuration.
- c. Restart applications associated with the server.

## Configure agent keys

Agent keys let agents communicate with Contrast. Contrast supports the ability to configure multiple agent keys with different authentication credentials instead of sharing one key across an organization.

The benefits of having multiple agent keys include:

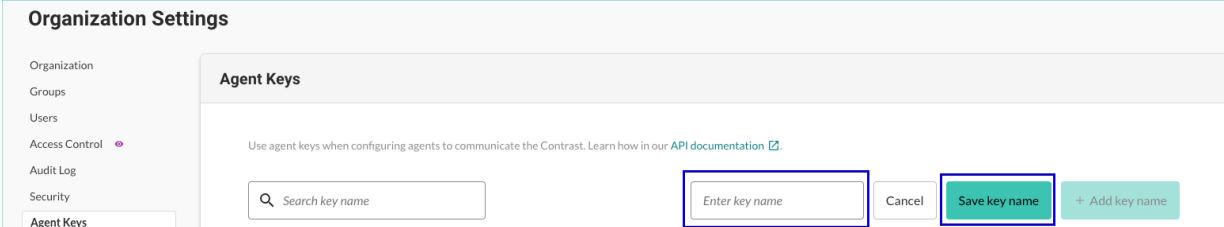
- **Enhanced security:** Each team or service can manage its own credentials, reducing the risk of a single point of failure.
- **Operational flexibility:** Teams can delete keys, add keys, or manage access without impacting other teams or services within the organization.
- **Granular control:** By assigning specific agent keys to different contexts, organizations can better control and monitor agent access and activity.

## Before you begin

- To add or delete agent keys, you need these actions or roles:
  - If you are using role-based access control, you need a role with the Edit organization action.
  - If you are using organization users and groups, you need the Organization Admin role.
- To view agent keys, you need these actions or roles:
  - If you are using role-based access control, you need a role with the Manage organization rules or Edit organization action.
  - If you are using organization users and groups, you need the Rules Admin or Edit role.

## Add an agent key

1. Under the user menu in the Contrast web interface, select **Organization settings**.
2. Select **Agent keys**.
3. Select **Add key name**.
4. Enter a key name at the top of the list.
5. Select **Save key name**.  
Contrast displays the keys for the new key name.



## Delete an agent key

When you delete an agent key, any active agent using it will be unable to authenticate and will shut itself down.

1. Under the user menu in the Contrast web interface, select **Organization settings**.
2. Select **Agent keys**.
3. Locate the agent key name you want to delete.
4. Select the **Delete** icon (🗑️) at the end of the row.
5. When prompted to do so, confirm you want to delete the key name by selecting **Delete**.

## Manage users, groups and permissions at an organization level

If [allowed at a system level \(page 1221\)](#), an Organization Administrator can manage user permissions for the organization by:

- [Adding a user or editing user settings \(page 1163\)](#)
- Managing organization permissions with [access groups \(page 1164\)](#)

## Add or edit a user at an organization level

If you add users within an organization, you can assign them to access groups for the applications within that organization. For most Contrast installations, the [default \(page 1221\)](#) is to set roles and permissions at an organization level, however if you have users who need to access multiple organizations, [add those users at a system level \(page 1221\)](#).

To add users:

1. Log in to Contrast with Organization Administrator permissions.
2. Select **Organization settings** in the **user menu**.

3. Select **Users** in the left navigation.
4. Select a user name from the list of users to edit their entry, or select **Add user** to add a new user.
5. For each user you can enter:
  - **Organization role:** Select one of the default [organization roles \(page 1267\)](#) that apply to all applications in this organization or [create a custom access group \(page 1164\)](#).
  - **Application access groups:** Select one of the default [application roles \(page 1264\)](#) that apply to all applications in this organization or create a custom access group to grant specific permissions to certain applications.
  - **Access permissions:** You can allow users access to the API, to the Contrast web interface and to Protect data. (Protect permissions can also [be granted at a system level \(page 1226\)](#).)

**TIP**

If you assign someone an administrator role, be sure to grant them both API and Contrast web interface access.

6. Select **Add** or **Save**.

### Add, edit, or delete an organization access group

Use organization access groups to assign users permissions and capabilities by [role \(page 1264\)](#).

Contrast provides default access groups that you can use instead of creating your own:

- **View:** Members of this group have read-only access to the Contrast interface to see scores, libraries, vulnerabilities and comments.
- **Edit:** Members of this group can remediate findings, add tags, manage vulnerabilities, edit attributes, merge applications, add or delete applications, and create servers.
- **Rules Admin:** Members of this group can edit rules and policies in the application, enable Protect, manage notifications and scoring.
- **Admin:** Members of this group can configure and manage settings for the organization.

### Before you begin

An Organization Administrator role is required.

### Steps

1. Under [organization settings \(page 1158\)](#), select **Groups**.
2. Select an existing group to edit, or select **Add group** to create a new group.

**TIP**

To find groups you can use the quick filter dropdown or the search field in the top left, or use the up/down arrows at the top of each column to sort.

The default groups that Contrast provides, indicated with a lock icon, have fixed applications and roles, and can't be deleted. You can only add or remove users from these default groups.

3. Fill out the form with:
  - **Group name:** Choose something that reflects the purpose, permissions and capabilities you will assign to this group.
  - **Application access:** Select the application name here to associate this group with the application. You can also set a group name when you are setting up a new application.



- **Role:** Select the application role you want the members of this group to have within the corresponding application.
  - Select **Add access** to add more applications and roles.
4. Next to **Members**, on the right, type ahead to select one or more users to assign to the group.
  5. When you are finished, select **Add** to create the new group.

**NOTE**

If users are assigned to two groups with conflicting roles for all applications or organizations, the role that provides the most restrictive access applies.

6. To delete a group, select **User menu > Organization settings > Groups**. Find the group you want to delete and select the Delete icon in that row.  
Once this is confirmed, the group is removed and any access provided by that group is revoked from all users assigned to the group.

**TIP**

To assign a user a role for all applications in the organization, assign them both an organization role and an application role from the default role groups. (For example, set both the organization and application roles to "Administrator" and they will have administrator permissions for all applications in your organization.)

To give a user access to a particular application, create an access group for that application and add the user to that group. Users not assigned to any application access groups won't have access. A user can have various roles across applications within a single organization.

Most Contrast customers use single organization deployments. Groups created at an organization level impact the roles and permissions across that particular organization. Organization access groups can also be created [at a system level \(page 1224\)](#) to allow users access to more than one organization

## Create an API only user

Create an API Only user account that you can use for all plugins or integrations.

**NOTE**

This procedure is for on-premises customers only.

If you are a hosted customer use this [Create an API only user \(page 1307\)](#) procedure.

**Best practice:** Add a user account that's only purpose is for use with plugins and integrations. Doing so avoids a situation where a user leaves and you delete that user's account. The deletion of that account would result in breaking the plugins and integrations that you use.

An API only account does not receive email notifications, even if the notification settings are turned on.

## Before you begin

- An Organization Admin role is required.
- API Only users can access Contrast's REST API but cannot log in to the Contrast web interface.
- If you configured your organization to use SAML-based single sign-on (SSO), you can still create an API only user.

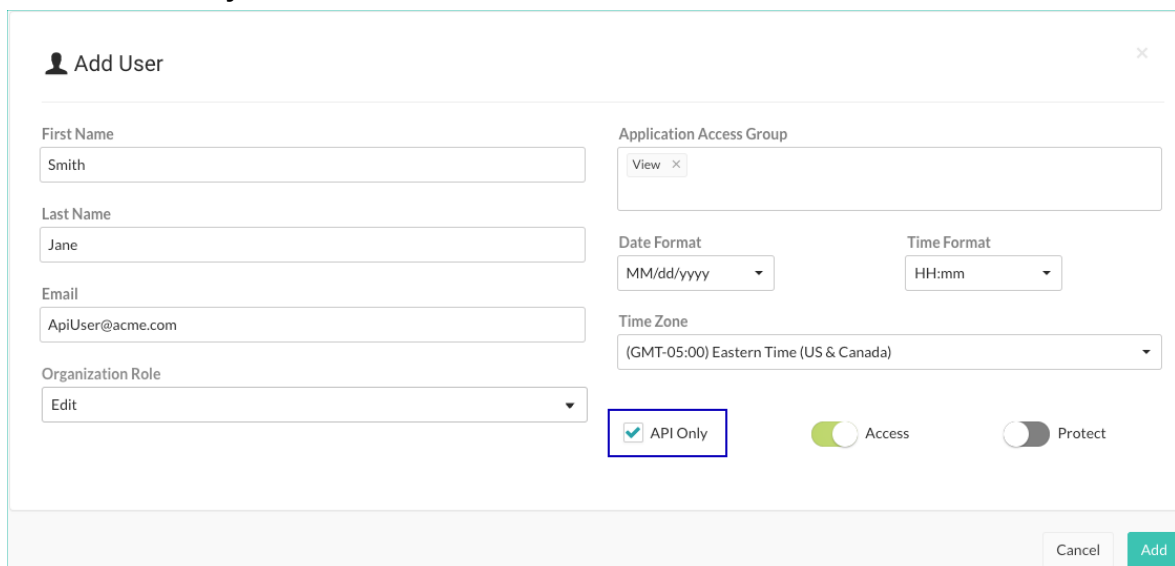
## Steps

To create an API only user:

1. From the user menu, select **Organization settings**.
2. Select **Users**.
3. Select **Add User**.

 + Add User

4. Enter the user name , email address, and time zone information.
5. Select the **Role**.  
**Best practice:** Select **Edit** (page 1267) for the Organization role to give the user the least permissive role.  
It is not recommended to give API only users Admin permissions.
6. Select an Application access group.  
**Best practice:** Select **View** (page 1264) or **Edit** (page 1264) for the Application access group.  
Depending on the API endpoints you want to call, and if you are trying to GET (read) or POST (write), the API only user might require the higher Edit permission instead of View.
7. Select the **API only** checkbox.



### NOTE

Selecting the API only checkbox overrides the Access option, if it's enabled. API only users have no access to the Contrast web interface.

8. In **Organization Settings > Users**, verify that you can see with new user with the **API Only** label next to the name.

USERS ?							
All (1) sm							
Name (Username)	Status	Organization Role	Last Login	UI Access	Serverless	Protect	
Smith Jane ApiUser@acme.com	API Only	✓	Edit	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

9. If you are using access groups to restrict access to specific applications, add the API only user to the [groups \(page 1164\)](#) for the applications that you want the API user to access. Verify access by looking at **Permissions** in the user profile.
10. To use the API only user account, get the connection strings:
  - a. From the user menu, select **Organization settings**.
  - b. Select **Users**.
  - c. Hover over the API only label next to the user's name and copy the displayed Service key. The Service key is unique to each user. All other API keys are shared across all users.
  - d. Create the Authorization header with a command similar to this example:

```
echo -n '[email address of the API only account:Service Key]' |
base64
```

## Manage usage analytics

Contrast collects usage data to understand how to build a better product. Confidential customer data is not tracked, and all user data is anonymized for analytics performed on aggregate information.

As an Organization Administrator, you can enable or disable usage data collection for the organization:

1. Under [organization settings \(page 1158\)](#), select **Organization**.
2. Under **General settings**, use the toggle to disable or enable **Product usage analytics**. It is enabled by default.

Diagnostics are also [managed at a system level \(page 1250\)](#).

## Restricted edit role

If you want to restrict users from deleting vulnerabilities and archiving applications, use the Restricted edit role setting. When selected, this setting applies to all users with the Organization edit or Application edit role.



### NOTE

This setting is available for on-premises customers only.

## Steps

1. From the user menu, select **Organization settings**.
2. Select **Organization**.
3. Under General settings, select **Restricted edit role**.  
When selected, all users with the Organization edit or Application edit role cannot delete vulnerabilities or archive applications.

## Set a password policy at an organization level

As long as it is [allowed at a system level \(page 1246\)](#), as an Organization Administrator, you can regulate passwords within your organization by creating a password policy.

1. Under [organization settings \(page 1158\)](#), select **Security**.
2. Enter the following settings for your policy:
  - Password Strength: This can be **Weak**, **Medium**, **Strong**, **Complex** or **Custom**. If you choose **Custom**, enter the minimum required number of **Uppercase letters**, **Lowercase letters**, **Numbers** and **Symbols**.
  - Enter the number of characters required in the **Minimum length** field.
  - Use the dropdown to choose the length of time allowed before **Password expiration**.
  - Enter the number of login attempts allowed before **Login logout**.
  - Choose the length of time allowed before **Inactive account expiration**.
  - Check the box to **Restrict password reuse**, and use the dropdown to choose the number of times each password may be reused.
  - Check the box to **Restrict password reset**, and use the dropdown to choose the number of hours during which a user can reset their password after their reset request is sent.
  - Use the dropdowns to select the amount of time that may pass before **Idle timeout** and **Session timeout**.
3. Select **Save**.

## Restrict email domains

Restrict which email address domains can receive Contrast information.

1. Under organization settings, select **Security**.
2. Under **Email domains**, enter comma delimited email domain names that should be allowed to receive information from Contrast. For example, `yourbusiness.com`.
3. Select **Save**.

## Set IP range

Restrict which IP addresses can access your Contrast account. This affects both browser and API access.

1. Under [organization settings \(page 1158\)](#), select **Security**.
2. Enter the IP address for which you want to allow access. Select **Add IP address** to enter additional addresses. You can also specify a range of IP addresses by using Classless Inter Domain Routing (CIDR) notation `10.0.0.0/24`.
3. Select **Save**.

## Enable multi-factor authentication at an organization level



### NOTE

For on-premises customers, set multi-factor authentication [at a system level \(page 1232\)](#).

If a user belongs to multiple organizations, their default organization determines their multi-factor authentication settings.

## Before you begin



### NOTE

- If you configure SSO in Contrast and also want to use multi-factor authentication, configure it using your identity provider (IdP), instead of Contrast. With SSO configured, Contrast passes the responsibility of authenticating users to the IdP.
- If you're not using SSO, protect your system by requiring [multi-factor authentication \(page 1168\)](#) as a required setting.

## Steps

1. Under [organization settings \(page 1158\)](#), select **Security** in the left navigation.
2. Toggle the MFA setting to green.
3. Specify whether to require MFA for users in the organization.  
Multi-factor authentication is enabled and users can [choose how they want to receive multi-factor authentication notices. \(page 597\)](#)

## Configure single sign-on (SSO) at an organization level

For on-premises customers, single sign-on can be [configured at a system level \(page 1242\)](#). For hosted customers, Contrast Security configures authentication; however, a user may be granted permissions to set up SSO for their organization.



### NOTE

- If users are identified with a user ID rather than an email address, those accounts don't automatically transfer over to the SSO configuration and must be recreated.
- If you configure SSO in Contrast and also want to use multi-factor authentication, configure it by using your identity provider (IdP), instead of Contrast. With SSO configured, Contrast passes the responsibility of authenticating users to the IdP.
- If you're not using SSO, protect your system by requiring [multi-factor authentication \(page 1168\)](#) as a required setting.

1. From the user menu, under Organization settings, select **Single-sign-on** and select **Get started**. To change an existing configuration, select **Edit** at the bottom of the screen.
2. You may receive a warning window regarding the implications of changing authentication. Read it carefully before proceeding.
3. Use the displayed information at the top of the screen to set up Contrast with your IdP.

Organization

Groups

Users

Access Control

Security

Agent

Single Sign-On

Integrations

Servers

Applications

Notifications

Score Settings

SINGLE SIGN-ON

What you'll need: [Copy Metadata URL](#)

[Revert to Contrast-Managed Authentication](#)

Entity ID *	Metadata URL	Attributes
https://mycompany.com/Contrast/saml/metadata	https://mycompany.com/Contrast/saml/metadata	None required
Version	NameID Format	Consumer Binding
2.0	Email Address	HTTP-POST
Assertion Consumer URL		
https://mycompany.com/Contrast/saml/SSO		

What we need:

Identity Provider \*

My/Company Okta

☒ I have access to the metadata URL

IdP Metadata URL \*

https://mycompany.okta.com/app/sso/saml/metadata

Accepted Domain(s) (New Users) \*

mycompany.com

[+ Add domain](#)

Default Organization Role (New Users) ?

View

Default Application Access Group ?

View

Default Organization User Access Group ?

ADMIN UAG

☒ Enable user provisioning ?

☐ Add users to their Contrast groups upon SSO login ?

☐ Remove users from their Contrast groups upon SSO login ?

[Cancel](#) [Save](#)

4. Provide a name for your IdP as well as the associated metadata to connect to Contrast.
5. If you want to automatically create new user accounts when someone make a SAML request to log in to Contrast, select the **Enable user provisioning** checkbox at the bottom of the screen.
  - a. Add the **Accepted domains** that must be used to trigger user provisioning (for example, contrastsecurity.com)
  - b. Specify roles and groups:
    - **On-premises customers:** Use the dropdowns to choose the **Default organization role** and **Default application access group** for the new users.  
The value displayed for the Default User Access group does not apply to your configuration.
    - **Hosted customers:** If role-based access control is turned on, use the dropdown to select a **Default User access group** for the new users.  
The values displayed in the Default organization role and Default application access group do not apply to your configuration. If you are using the legacy access control, follow the instructions for on-premises customers.
6. If you want to add users who are members of a SAML group that matches the name of a Contrast group, select **Add users to their Contrast groups upon SSO login**.
7. If you want to remove users who are members of a SAML group that *does not* match the name of a Contrast group, select **Remove users from their Contrast groups upon SSO login**.
8. Select **Save**.
9. Open a **new** browser window, private browsing session or incognito window, and attempt the SSO login with your account. If you're unsuccessful, go back to the browser in which you're still logged in, disable SSO for the organization, select **Revert to Contrast-managed authentication** and confirm the change.

## See also

- [Configuring user and group provisioning with Okta](#)
- [Configuring ADFS to automatically add users to groups](#)

## Set server defaults at an organization level

Server settings provide default configurations to new servers (and their agents) that you add to Contrast. You can customize these configurations and set specific defaults for each environment.

### Steps

1. Under [organization settings \(page 1158\)](#), select **Servers**.
2. Use the dropdown to choose the environment in which you want to apply the default (Development, QA, or Production). Check the box next to **Set as default environment** if you want to specify a default environment for future server configuration.
3. Use the dropdown to choose the **Log Level**. The default [log level \(page 1210\)](#) selection is **ERROR**.
4. Under **Automatic server cleanup**, enter the length of time that you would like servers to be offline before they are automatically cleaned up. The default value is 30 days.  
A background task runs every five minutes to check if there is an organization with automatic server cleanup enabled.  
If there are one or more servers with no activity received within the configured time frame, Contrast disables the servers automatically. They are no longer visible under **Servers** in the Contrast web interface.  
Contrast keeps Information on vulnerabilities and attacks related to these servers even after they're disabled. Protect licenses from disabled servers return to the pool of licenses.
5. Under **Assess**, specify these settings:
  - a. Select which stack traces should be captured (all, some or none).
  - b. To optimize analysis performance, select **Enable sampling for higher performance**.
    - If Contrast sees the same URL being called multiple times, it analyzes the URL based on the the number of times specified in the Baseline setting.
    - Afterwards, if Contrast continues to see the same URL, it only checks it based on the Frequency setting.
    - Contrast retains samples for the number of seconds specified for the Window setting. After the time specified for the Window setting elapses, Contrast analyzes the URL again, according to the Baseline setting.Configure these settings:
  - **Baseline:** The number of times that Contrast analyzes URLs to complete sampling. The default setting is **5**.
  - **Frequency:** The number of times that Contrast analyzes URLs after the Baseline is achieved. The default setting is **10**.
  - **Window:** The number of seconds that Contrast retains samples before reverting to the Baseline. The default setting is **180**.
6. Under **Protect**, specify these settings:
  - a. To enable Protect, turn on the Protect toggle.



### IMPORTANT

Turning Protect on selects the setting to apply Protect licenses to new servers automatically.

Administrators receive emails each time a server is licensed. As servers go up and down frequently, you may want to setup an email filter for any unwanted traffic.

In this section, the license bar shows the number of purchased Protect licenses in use. If you are using more licenses than you purchased, the license bar also shows the number of additional licenses in use.

- b. To turn on bot blocking, select **Enable bot blocking**. Bot blocking blocks traffic from scrapers, attack tools and other unwanted automation. To view blocked bot activity, under **Attacks > Attack Events**, use the **Automated** filter option.

**NOTE**

You can configure bot blocking in the [YAML files \(page 107\)](#) for Java, .NET Framework, .NET Core, Ruby, and Python.

- c. To send Protect events to syslog, select **Enable output of Protect events to syslog**. Configure these settings:
  - Enter the **IP Address** and **Port** in the given fields. Use the dropdown to choose the **Facility**.
  - Click on the event severity badges, and use the dropdown to choose a message **Severity** level for each one. The defaults are:
    - **1 - Alert** for Exploited
    - **4 - Warning** for Blocked
    - **5 - Notice** for Probe
7. To retain library details, turn on **Retain Library Data**. When enabled, Contrast retains library details for the last server being deleted from Contrast during server cleanup.
8. To send agent data to Contrast, turn on **Agent diagnostics** toggle. Contrast uses this data to improve rules, performance, and to prioritize product improvements.
9. To remove routes that are seen only on servers that are no longer active, turn on **Enhanced server cleanup**. [Configure enhanced server cleanup \(page 1172\)](#) provides details about this setting.

## Configure enhanced server cleanup

The Enhanced server cleanup setting lets you remove routes from the historical routes dataset if those routes are only seen on servers that are no longer active. This feature is an alternative to [route expiration \(page 626\)](#) policies.

Enhanced server cleanup is useful primarily if you have ephemeral servers and prefer to manage Assess results as unique, point-in-time scans of the application. For example, it could be beneficial if you use one server for one test run of a Contrast agent-instrumented application and don't expect to test run that application again for weeks or months.

If, as a user of Assess and Application Vulnerability Monitoring (AVM), you continuously instrument your applications, use a route expiration policy instead. It allows for more flexibility and control over when stale routes are cleaned up, regardless of the status of the server the route was reported on. Route expiration is also a better choice for if you have long-running servers, as they are more likely to host multiple versions of the same application.

### Additional benefits

The benefits of using this setting include:

- Remove routes that may no longer be seen in current builds of the application
- An improvement in route coverage percentage
- An improvement in vulnerability burn down metrics
- Less data in the Route Coverage tab

### How enhanced server cleanup works

When you set a server cleanup policy and turn on enhanced server cleanup, Contrast:

- Removes all servers except the last active server  
Contrast keeps the most recently active server for each application, even if all servers for an application are offline.



- Remove routes that may no longer be seen in current builds of the application
- Removes routes that are associated with no servers when the server is removed
- Marks vulnerabilities that are associated with no servers as **Verified- Auto-remediated** when the server is removed
- Updates the total count of routes and the denominator in route coverage
- Updates total count of vulnerabilities based on the auto-verified status

## Before you begin

- The enhanced server cleanup setting affects all server environments.
- Use the Contrast web interface or the [Contrast API](#) to configure this setting.

## Steps

1. From the user menu, select **Organization settings**.
2. Select **Servers**.
3. Under Automatic server cleanup, set the server cleanup policy by entering the length of time that you would like servers to be offline before they are automatically cleaned up.  
Any Protect licenses assigned to removed servers are returned to the pool of available Protect licenses.
4. Select **Enhanced server cleanup**.
5. Select **Save**.

**Retain Library Data** ☐ **!** Applies to all server environments.  
Libraries will still be visible for applications on deleted servers

**Agent Diagnostics** ☒ **!** Applies to all server environments.  
Send Contrast agent data to improve rules, performance and to prioritize product improvements.

**Enhanced Server Cleanup** ☒ **!** Applies to all server environments.  
Enhances automatic server cleanup to remove routes and mark vulnerabilities as auto-remediated when those routes or vulnerabilities were only seen on servers that are being cleaned up.

Save

## Set application defaults at an organization level

Use this procedure to choose default settings for applications at an organization level.

## Before you begin

- An Organization Admin role is required.

## Steps

1. Under [Organization settings \(page 1158\)](#), select **Applications**.
2. Use the dropdown to choose an **Importance** level for applications. Your options are **Critical**, **High**, **Medium**, **Low**, **Unimportant**. The default selection is Medium.
3. Under Policy, select the remediation and [compliance policies \(page 1147\)](#) you would like to automatically apply to applications.  
You can still add applications to policies that aren't included in your default settings at a later time.
4. Under Application onboarding, select either or both of these settings:

- Do not onboard applications that are missing application metadata
- Do not onboard applications that are missing session metadata

Depending on the settings you select, Contrast fails to add applications that are missing required metadata. These settings help you to enforce the metadata configuration in the agent's configuration file.

5. Under Session metadata, to set the default [session metadata filtering \(page 616\)](#) to the most recent session, select **Filter application details by most recent session**.  
This selected filter affects the application's Vulnerabilities and Route coverage tabs.
6. Under Behavior, select **Automatically apply licenses to new applications** if you want licenses to be applied automatically.  
A status bar shows you how many licenses have been used out of the total number available.  
Select the license link to understand the breakdown of your organization's licenses.
7. Use the Application metadata section to [configure application metadata \(page 1174\)](#) that should be provided for each of the applications in your organization.
8. Select **Save**.

### Create requests for application metadata

You can configure requests for application metadata that is collected whenever you add a new application to Contrast.

When you [install and configure an agent \(page 52\)](#), you are prompted to enter metadata for the fields you create and to add the information in the agent configuration file. The metadata is then displayed in the **Applications** list, where you can also use it to filter applications, and the application's **Details** page in the Contrast web interface.



#### NOTE

The following agent versions support application metadata fields:

- Java 3.5.6.591 and later
- .NET 18.10.35 and later
- Node 1.35.0
- Python 1.2.0
- Ruby 2.0.8



#### IMPORTANT

The data supplied for application metadata is required for Agent configuration and YAML file download. The downloads will be disabled without the data.

### Steps

1. Under organization settings, select **Applications**.
2. Under Application metadata, for each field enter:
  - **Field type:** Freeform, Numeric or Point of contact. The type of field determines the type of validation.
  - **Name:** Enter a label for this field.

To ensure compatibility with the Contrast APIs, use lower camel-case formatting for application metadata names. For example, use `businessID`, not `BusinessID` or `Business ID`.

- **Value condition:** Use the checkbox to indicate whether the metadata value provided should be **Required** or **Unique**.

3. Select **Add field** to complete as many rows as needed.
4. As you provide information for each field, you will see the formatted property that you can copy and paste into your agent configuration files. Add the information for each key=value pair to the agent configuration file.
5. To prevent reporting of data for applications that don't include all required fields, select **Restrict applications missing required fields**. This option applies to new and existing applications in the organization.

When you select this option, the Contrast web interface displays a warning message if an application is missing a required field in the agent configuration file. The Contrast web interface displays the application, however, the agent reports no data for it, including exercised routes and vulnerabilities.

If you choose not to restrict applications, any application missing a required field is successfully added and the agent reports data. Contrast displays a warning message that one or more fields are missing.

## Manage notifications at an organization level

Notifications alert users in specific situations, such as the discovery of a vulnerability or an attack on an application. Organization Administrators can [set default settings \(page 1176\)](#) for Contrast notifications for all users in their organization. Individual users can [choose how they want to receive these notifications \(page 599\)](#).

There are two primary channels available for notifications:

- **In Contrast:** Notifications are available directly in the Contrast application. Select the **Notification** icon in the top right header to view your notifications.
- **Email:** A System Administrator can [configure Contrast \(page 1254\)](#) to communicate with an appropriate SMTP system to receive notifications by email.



### NOTE

For some features that require user notifications, Contrast automatically notifies the affected users when a Contrast administrator enables the feature. (You can't control these notifications in the **Notifications** page.) Contrast requires user and administrator notifications for features including [vulnerability status approval \(page 1178\)](#) and other **Policy Management** settings.

## Set administrative notifications

Administrators automatically receive the following notifications for high-level events in their organization in the Contrast application and by email.

- **Application licensed:** A new application was [licensed \(page 1161\)](#) in Contrast.
- **Application license expiring:** The license for an active application is expiring. (Contrast sends this notification two months, one month and one week prior to the expiration date).
- **Licenses expiring:** Existing license(s) with no associated applications is expiring. (Contrast sends this notification two months, one month and one week prior to the expiration date).
- **Remediation policy violation:** A vulnerability is in violation of an existing [remediation policy \(page 1124\)](#)

- **Library policy violation:** A library is in violation of an existing [library policy \(page 1151\)](#).
- **Compliance policy violation:** An application is in violation of an existing [compliance policy \(page 1147\)](#).

Administrator and RulesAdmin users at the application and the organization level must receive policy violation notifications. You can [manage your notifications \(page 599\)](#) to minimize the number of notifications or consolidate them into a single email.

You can also receive notifications soliciting approval when a user requests to close certain vulnerabilities. This must be [configured \(page 1178\)](#) by an Organization Administrator.

## Set default user notifications

As an Organization Administrator, you can define default notification settings for all users in your organization both for integrations and for email and in-app notifications in Contrast. Individual users can [choose how they receive those notifications \(page 599\)](#).

### Steps

1. Under [organization settings \(page 1158\)](#), select **Notifications**.
2. Use the toggles to enable or disable the subscriptions listed on the left:
  - **Active attack:** There is an active attack on an application with Protect enabled.
  - **New vulnerability:** Contrast has detected a new vulnerability. Click in the field to enable notifications for specific severity levels; the default selection is **All**.
  - **Vulnerable library:** Contrast detected a new vulnerable library or a new CVE in a library.
  - **Server Messages:** Receive messages about reliability issues for servers.
  - **Server offline:** Contrast can't reach a server.
  - **New comment:** A team member commented on a finding.
  - **New asset:** A new asset to which the user has access was onboarded. Click in the field to set this notification for **Application** or **Server**; the default selection is **All**.
  - **Nearing expiration:** An application license is about to expire.
  - **Policy violations:** A vulnerability is in violation of a compliance, policy, or remediation policy. To receive daily email digests of violation notifications, select **Aggregate policy violation emails into separate daily email digests** and select the type of policy notifications you want to include in the email.
  - **Scan failed:** A Contrast scan failed. The notification specifies the name of the Scan project that contains the failed scan.
  - **Email digest:** A daily summary of Contrast activities. (Email only)

## Enable subscription notifications for integrations

To enable subscriptions for a particular integration, select **Add integration** to add an integration or select an existing integration from the dropdown at the top of the Integrations column.

## Create custom notifications

As an Organization Administrator, you can set notification defaults, or create custom notifications.

To create a custom notification:

Select **Organization settings > Notifications > Create notification** at the top of the **Custom notifications** list. In the window that appears, fill out the following form fields.

1. Use the radio buttons to choose **Vulnerability** or **Attack**.
2. Choose a **Name** for the notification.
3. Use the dropdown to set the notification **Interval** as **Daily**, **Weekly**, or **On Event**.
4. Enter a **Description** for the notification's purpose.

- Click in the multiselect field to choose the **Applications** for which this notification applies.
- Choose the **Application Tags** for which this notification applies.
- Choose which organization **Users** should receive the notifications.
- Use the dropdowns to choose your **Conditions**.

Click the **Add Condition** link to add a row.

Contrast supports these conditions for custom notifications:

Notification types	Condition	Description
Category	Is or Is Not	Categories are high-level groupings of rule types such as Authentication, Injection, Cryptography, etc. There are 11 categories within Contrast rule types.
Impact	Is, Is Lower Than, Is Higher Than	Impact is measured in High, Medium and Low ratings based on how a rule type affects a given organization. Every rule type has a default impact configuration setting which can be customized.
Likelihood	Is, Is Lower Than, Is Higher Than	Likelihood is measured in High, Medium and Low ratings based on how frequent a rule type may occur. Every rule type has a default likelihood configuration setting that can be customized.
URL	Is, Contains, Starts With	A specific URL from an application.
Class	Is, Contains, Starts With	A specific Java or .NET class.
Method	Is, Contains, Starts With	A specific Java or .NET method.



### IMPORTANT

If you choose multiple conditions, Contrast uses AND logic for the notifications. Contrast generates the notifications when all selected conditions apply to the situation.

## Customize score settings at an organization level

Contrast designates an [application score \(page 1269\)](#), which can optionally depend on a [library score \(page 934\)](#). To customize score settings at an organization level:

- Under [organization settings \(page 1158\)](#), select **Score settings**.
- Under **Overall score**, choose how applications in this organization are scored:
  - Default score** is the average of your application's library score and its custom code score.
  - Custom code-only score** ignores library score when calculating the overall application score. If you select this option, you can click to select specific languages, or apply it to all languages.
- Under **Library score**, choose how libraries in this application are scored:
  - Default score** uses an algorithm that includes vulnerabilities, as well as the age and versioning of a library.
  - Vulnerability-only score** bases scoring solely on vulnerabilities present in the library.
- Select **Save**.



### TIP

A RulesAdmin can configure policy settings in **Policy Management** so that any library in violation automatically receives a failing score (F). Once these settings are chosen, you'll see an alert message in Score Settings. Clicking the policy link in the alert navigates you to Library Policy, where administrators may view and revise these settings.

## Require vulnerability approval

As an Organization Administrator, you can [require administrative approval when closing vulnerabilities \(page 1124\)](#) in your organization. You must be an Organization RulesAdmin with RulesAdmin permissions for the target application in order to [approve or deny vulnerability closures \(page 1038\)](#).

To configure this requirement:

1. In the **user menu**, select **Policy management > Vulnerability management > Vulnerability behavior**.
2. Select the box next to **Require administrator approval when closing vulnerabilities**.
3. Choose the statuses and severities of vulnerabilities that should automatically go into a **Pending** state when a user moves to close them.
4. When a user requests to close any qualifying vulnerabilities, Contrast sends an in-app notification to all Organization Administrators saying that a review is needed.  
Each vulnerability status will remain **Pending** until an Organization Administrator submits a review of the closure. To qualify for administrative approval, *both* a status and severity must be selected. If a reviewer denies the closure of a vulnerability, they must provide a reason for denial. Once confirmed, the reviewer's feedback appears in the vulnerability's **Activity** tab.  
If you disable the feature, any pending closures are automatically approved.



### NOTE

While in a **Pending** state, the vulnerability's previous status still applies for the purpose of organizational reports and statistics.

## View audit log (hosted customers) ☹

Contrast captures activity about user sessions including changes to settings or licenses, actions on vulnerabilities, and much more.

### Before you begin

- This feature is supported for hosted customers only. On-premises customers should visit [View audit log \(page 1180\)](#).
- If you are using [role-based access control \(page 1277\)](#), the View audit logs action is required. Role-based access control is part of the Contrast pre-release customer testing program. If you want access to this feature, contact your Contrast representative.
- If you are using [users and groups \(page 1163\)](#) for access control, an Organization Admin role is required.

### Steps

1. From the user menu, select **Organization settings**.
2. Select **Audit log**.
3. Search for a message or specify a date range.  
You can search for text or a specific date.

**Organization Settings**

Organization  
Groups  
Users  
Access Control  
**Audit Log**  
Security  
Agent Keys

Last week

GENERATED ON	MESSAGE
2024/16/10 10:55 AM	auto-test-user@ cations [ ] .vulne 271b6}. Justifica

## Audit log details

The audit log contains these details:

Alerts	Creating Deleting Updating
Applications	Applying a license Archiving Changing Assess rule configurations Changing organization settings Enabling/disabling a rule Merging/unmerging Restoring Resetting
Attacks	Adding notes to events Creating/Editing/Deleting notes Suppressing
Bugtrackers	Creating Updating
Email	Adding domains Sharing libraries
Groups	Creating/Updating/Modifying membership of organization groups
IP ranges	Adding Removing
Keys	Rotating API and Agent service keys
Notification settings	Updating for an organization or user
Patches	Creating/Updating/Toggling/Deleting virtual patches
Policies	Changing cleanup settings Changing library restrictions Changing password Changing scoring Changing timeouts Creating/Deleting/Disabling/Enabling/Updating compliance policies Creating/Deleting/Updating/Enabling organization remediation policies Creating/Deleting/Updating/Enabling security controls Creating/Deleting/Updating/Enabling rule exclusion policies

Reports	<p>Creating reports</p> <p>Vulnerability trend reports</p> <p>Downloading the PDF file for the Aggregated Vulnerability report</p>
<p>Role-based access control</p> <p>Role-based access control is a preview feature and not turned on for all users.</p>	<p>Creating/Updating/Deleting users</p> <p>Creating/Updating/Deleting resource groups</p> <p>Creating/Updating/Deleting roles: Includes updates to built-in roles.</p> <p>Creating/Updating/Deleting user access groups</p>
Scan	<p>Creating/Deleting scan projects</p> <p>Running a scan</p> <p>Changing the status of a vulnerability</p>
Servers	<p>Changing defaults</p> <p>Creating notifications</p> <p>Deleting/Disabling/Updating/Removing licenses</p> <p>Enabling/Disabling Protect</p>
SuperAdmin	<p>Creating</p> <p>Impersonating a user</p> <p>Updating</p>
Traces	<p>Adding/Editing/Deleting notes</p> <p>Auto-remediation</p> <p>Sharing/Deleting/Merging/Marking status of a trace (or bulk traces)</p> <p>Updating severities</p>
Users	<p>Adding</p> <p>Activating</p> <p>Changing access</p> <p>Creating/creation via provisioning/in bulk</p> <p>Deleting</p> <p>Granting/Revoking Protect</p> <p>Importing into an organization</p> <p>Underprivileged user attempts</p> <p>Updating</p>
Webhooks	<p>Creating</p>

## View audit log

Contrast captures activity about user sessions including changes to settings or licenses, actions on vulnerabilities, and much more.

This view is for on-premises customers. If you are a hosted customer, visit [View audit log \(hosted customers\) \(page 1178\)](#).

## Steps

1. Under the **User Menu**, select **Organization Settings > Security**.
2. Select **View Audit Log** on the upper-right of the page.



Contrast captures activity about all user sessions. [View Audit Log](#)



- Use the search fields to look for a specific log by date or name.

**Organization Settings**

- Organization
- Groups
- Users
- Security**
- API
- Single Sign-On
- Integrations
- Servers
- Applications
- Notifications
- Report Settings
- Score Settings

**Audit Log**

Contrast captures activity about all user sessions including changes to settings or licenses, actions on vulnerabilities, and much more

Find Audit Log

Start Date

End Date

Q Search

40 of 57595 events

Generated On	Message
01/20/2022 14:51	User i@contrastsecurity.com updated their account: [language=ja] differs from [language=en]
01/20/2022 14:46	- User i@contrastsecurity.com successfully logged in from [99. .31].
01/20/2022 14:06	- @contrastsecurity.com at [99. .31] was logged out due to inactivity.
01/20/2022 13:35	User updated their account: [language=en] differs from [language=ja]
01/20/2022 13:29	- User successfully logged in from [75. .150].
01/20/2022 12:38	- @contrastsecurity.com at [75. .4] was logged out due to inactivity.
01/20/2022 12:37	t@contrastsecurity.com impersonating @contrastsecurity.com - User t@contrastsecurity.com successfully logged in from [109. .37].
01/20/2022 12:37	- User @contrastsecurity.com successfully logged in from [73. .16].
01/20/2022 12:08	- User @contrastsecurity.com successfully logged in from [75. .4].
01/20/2022 11:43	- User @contrastsecurity.com successfully logged in from [73. .16].
01/20/2022 10:47	Remediation Policy 'test' disabled by @contrastsecurity.com
01/20/2022 10:47	Remediation Policy 'test' enabled by @contrastsecurity.com
01/20/2022 10:36	@contrastsecurity.com - @contrastsecurity.com - User @contrastsecurity.com successfully logged in from [109. .37].
01/20/2022 10:26	- @contrastsecurity.com at [73. .16] was logged out due to inactivity.
01/20/2022 10:13	- User @contrastsecurity.com successfully logged in from [75. .4].
01/20/2022 10:12	- User @contrastsecurity.com successfully logged in from [75. .4].
01/20/2022 09:52	@contrastsecurity.com impersonating @contrastsecurity.com] User i@contrastsecurity.com is now impersonating user @contrastsecurity.com
01/20/2022 09:51	@contrastsecurity.com impersonating @contrastsecurity.com] User @contrastsecurity.com is now impersonating user @contrastsecurity.com
01/20/2022 09:48	@contrastsecurity.com @contrastsecurity.com - User @contrastsecurity.com successfully logged in from [109. .37].
01/20/2022 09:47	@contrastsecurity.com @contrastsecurity.com - User @contrastsecurity.com successfully logged in from [109. .37].
01/20/2022 08:45	- User @contrastsecurity.com successfully logged in from [31. .58].
01/20/2022 08:36	i@contrastsecurity.com impersonating @contrastsecurity.com] User @contrastsecurity.com is now impersonating user @contrastsecurity.com
01/20/2022 07:45	Server 'PLogan-1' is offline
01/20/2022 07:40	- User @contrastsecurity.com successfully logged in from [31. .58].

## Audit log details

The audit log contains these details:

Alerts	Creating Deleting Updating
--------	----------------------------------

Applications	<ul style="list-style-type: none"> <li>Applying a license</li> <li>Archiving</li> <li>Changing Assess rule configurations</li> <li>Changing organization settings</li> <li>Enabling/disabling a rule</li> <li>Merging/unmerging</li> <li>Restoring</li> <li>Resetting</li> </ul>
Attacks	<ul style="list-style-type: none"> <li>Adding notes to events</li> <li>Creating/Editing/Deleting notes</li> <li>Suppressing</li> </ul>
Bugtrackers	<ul style="list-style-type: none"> <li>Creating</li> <li>Updating</li> </ul>
Email	<ul style="list-style-type: none"> <li>Adding domains</li> <li>Sharing libraries</li> </ul>
Groups	Creating/Updating/Modifying membership of organization groups
IP ranges	<ul style="list-style-type: none"> <li>Adding</li> <li>Removing</li> </ul>
Keys	Rotating API and Agent service keys
Notification settings	Updating for an organization or user
Patches	Creating/Updating/Toggling/Deleting virtual patches
Policies	<ul style="list-style-type: none"> <li>Changing cleanup settings</li> <li>Changing library restrictions</li> <li>Changing password</li> <li>Changing scoring</li> <li>Changing timeouts</li> <li>Creating/Deleting/Disabling/Enabling/Updating compliance policies</li> <li>Creating/Deleting/Updating/Enabling organization remediation policies</li> <li>Creating/Deleting/Updating/Enabling security controls</li> <li>Creating/Deleting/Updating/Enabling rule exclusion policies</li> </ul>
Reports	<ul style="list-style-type: none"> <li>Creating reports</li> <li>Vulnerability trend reports</li> <li>Downloading the PDF file for the Aggregated Vulnerability report</li> </ul>
<p>Role-based access control</p> <p>Role-based access control is a preview feature and not turned on for all users.</p>	<ul style="list-style-type: none"> <li>Creating/Updating/Deleting users</li> <li>Creating/Updating/Deleting resource groups</li> <li>Creating/Updating/Deleting roles: Includes updates to built-in roles.</li> <li>Creating/Updating/Deleting user access groups</li> </ul>
Scan	<ul style="list-style-type: none"> <li>Creating/Deleting scan projects</li> <li>Running a scan</li> <li>Changing the status of a vulnerability</li> </ul>
Servers	<ul style="list-style-type: none"> <li>Changing defaults</li> <li>Creating notifications</li> <li>Deleting/Disabling/Updating/Removing licenses</li> <li>Enabling/Disabling Protect</li> </ul>

SuperAdmin	Creating Impersonating a user Updating
Traces	Adding/Editing/Deleting notes Auto-remediation Sharing/Deleting/Merging/Marking status of a trace (or bulk traces) Updating severities
Users	Adding Activating Changing access Creating/creation via provisioning/in bulk Deleting Granting/Revoking Protect Importing into an organization Underprivileged user attempts Updating
Webhooks	Creating

## Impersonation

Impersonation lets someone with a [system role \(page 1269\)](#) access an organization as an existing user to troubleshoot issues.

Contrast turns off impersonation for organizations after 24 hours, automatically.

For hosted customers, if the impersonation setting is not visible for your organization, contact [Contrast Support](#). On-premises customers can manage and use impersonation, as needed.

## Impersonation access

By default, the impersonation setting is enabled for all organizations. Organization Admins manage whether you can impersonate another user in an organization.

To change access to impersonation:

- SuperAdmins can turn on or turn off the **Can enable impersonation** setting for a specific organization by [editing the organization \(page 1219\)](#).  
This setting affects whether Organization Admins can see the Impersonation setting for an organization.
- Organization Admins can manage impersonation use by [editing an organization's Security settings. \(page 1184\)](#)

When you turn on impersonation:

- If you have a SuperAdmin role, you can use [impersonation \(page 1230\)](#) by selecting **Impersonate** for an organization on the Organizations page.  
You can impersonate the first Organization Admin for the organization.
- If you have a SuperAdmin role, you can select a user to impersonate on the Users page.
- If you have a SuperAdmin role, you can use [impersonation \(page 1230\)](#) by selecting **Impersonate** for an organization on the Organizations page.  
You can impersonate the first Organization Admin for the organization. You must have access to the organization.

## Audit log

The Contrast audit log shows impersonation activity including:

- When impersonation is turned on or off
- The organization where impersonation occurred
- The server key associated with an impersonation status change
- Rejected impersonation attempts

## Enable impersonation

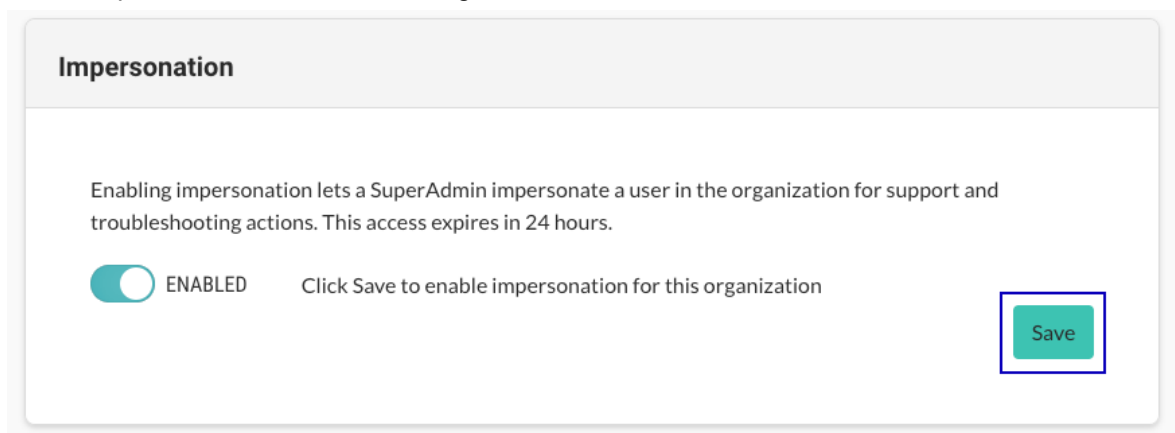
Use this procedure to turn on [impersonation \(page 1183\)](#) for an organization.

### Before you begin

- Verify that the setting to enable impersonation is visible under **Organization settings > Security**.  
If the setting is not visible:
  - **Hosted customers:** Contact [Contrast Support](#).
  - **On-premises customers:** Ask a SuperAdmin to [edit the organization \(page 1219\)](#) and turn on the **Can enable impersonation** setting.

### Steps

1. From the user menu, select **Organization settings**.
2. Select **Security**.
3. Under Impersonation, select the setting to enable it and select **Save**.



**Impersonation**

Enabling impersonation lets a SuperAdmin impersonate a user in the organization for support and troubleshooting actions. This access expires in 24 hours.

☒ ENABLED      Click Save to enable impersonation for this organization

[Save](#)

# System Administration

Only on-premises customers require system administration. System administration is handled by Contrast Security for hosted customers.

See an overview of how to [get started on-premises \(page 1185\)](#) or decide how to [manage your system administration \(page 1218\)](#).

## Get started on-premises

As an on-premises customer, you can set up your own instance of Contrast without a connection to the internet. You only need to set this up once per organization.



### IMPORTANT

If you are able to use Contrast as a hosted solution, you don't have to complete the additional installation and maintenance of the Contrast application on-premises.

Contrast hosted is SOC-2 Type II compliant and continuously receives feature updates. To connect to Contrast hosted, use the credentials provided by your administrator to log in and continue to [install an agent \(page 52\)](#).

For on-premises customers, to use Assess, Protect or both, you must complete at least two installations:

- [Install Contrast \(page 1190\)](#)
- [Install an agent for each application server \(page 52\)](#)

The Contrast installation contains all embedded components that make up the system configuration, including a Tomcat servlet container, MySQL database instance, and an AdoptOpenJDK Hotspot Java Virtual Machine. All of these components are embedded within the installation binary and deployed to a single server as part of the Contrast architecture.

Before installing the Contrast application, verify that your environment complies with the:

- [System requirements \(page 1186\)](#)
- [Sizing recommendations \(page 1187\)](#)

After installation, you can further configure:

- [Tomcat \(page 1202\)](#)
- [JRE \(page 1203\)](#)
- [HTTPS \(page 1203\)](#)
- [Contrast settings \(page 1248\)](#)

For the long term, make sure you have a plan to:

- [Manage system administration \(page 1218\)](#)
- [Configure settings \(page 1248\)](#)
- [Maintain the Contrast system \(page 1255\)](#)

## Contrast installation

Options for installing and deploying Contrast in an on-premises environments include:

- [Install with the Contrast installer \(page 1190\)](#)
- [Use a distributed MySQL database \(page 1194\)](#)
- [Deploy with a WAR file. \(page 1193\)](#)
- [Deploy in a distributed environment \(page 1197\)](#)

### Next steps

- Review the [Contrast system requirements. \(page 1186\)](#)
- Review the [Contrast sizing recommendations \(page 1187\)](#).
- Download the Contrast installer from the [Contrast Hub](#) or with [curl commands \(page 1188\)](#).

## Contrast system requirements

The following table lists the system requirements for installing the Contrast application.

Before you install Contrast:

- [Download the Contrast Installer \(page 1189\)](#) from the [Contrast Hub](#) or by using [curl commands \(page 1188\)](#).
- Review the [sizing recommendations. \(page 1187\)](#)

Requirement	Recommended	Minimum	Notes
OS Architecture	64-bit	64-bit	Due to memory requirements, the Contrast application can only run on 64-bit architectures.
Operating system	<ul style="list-style-type: none"><li>• RHEL 8</li><li>• RHEL/CentOS 7</li><li>• Microsoft Windows 2019</li><li>• Ubuntu 18.04 LTS or higher (up to 20.04 LTS)</li></ul>	<ul style="list-style-type: none"><li>• RHEL/CentOS 7</li><li>• Microsoft Windows 2012 R2 or higher</li><li>• Ubuntu 16.04 LTS</li></ul>	Support for CentOS 6 ended on December 1, 2020.
Java	Java is bundled with the installer.		
MySQL	<ul style="list-style-type: none"><li>• For on-premises customers using Contrast versions 3.8.11 or prior, MySQL 5.7.23 is bundled with the installer.</li><li>• Beginning with Contrast 3.9.0, MySQL 8.0 is bundled with the Linux on-premises installer.</li><li>• Beginning with Contrast 3.9.3, MySQL 8.0 is bundled with the Windows on-premises installer.</li><li>• Beginning with Contrast 3.12.1, MySQL 8.4 is bundled with the Linux and Windows on-premises installer.</li></ul> <p>If you experience issues, <a href="#">contact Support</a>.</p>		



### IMPORTANT

- For on-premises customers using MySQL 8.0 or later (which has binary logging enabled by default), the system variable `log_bin_trust_function_creators` must be set to **ON** so that Contrast can create stored procedures. For more details, see [MySQL documentation](#).
- For on-premises customers using MySQL 8.0 or later, the system variable `local_infile` must be set to **ON** so that Contrast can accept CSV files to help the import of SCA data. For more details, see [Security Considerations for LOAD DATA LOCAL](#).

## MySQL and Java requirements for distributed installations

Use these requirements if you are deploying Contrast as a distributed application or are using your own MySQL database. For all other on-premises installations, use the MySQL and Java software included in the Contrast installer.

If you experience issues, [contact Support](#)

Requirement	Recommended	Minimum
Java	17.0.0 - 17.0.x  Contrast displays a message if you try to use a version outside of this range.	17.0.0
MySQL	<ul style="list-style-type: none"><li>• 8.4 (Contrast 3.12.1 or later)</li><li>• 8.0.36 (Contrast 3.11.0 or later)</li><li>• 8.0.30 (Contrast 3.9.1 or later)</li><li>• 5.7.23 (Contrast 3.9.0 or earlier)</li></ul>	<ul style="list-style-type: none"><li>• 8.0.27 (Contrast 3.9.1 or later)</li><li>• 5.7.23 (Contrast 3.9.0 or earlier)</li></ul>

## SuperAdmin account

To ensure that connections for integrations work correctly, create a SuperAdmin account that is different from the default account that Contrast Security created. Continuing to use the default SuperAdmin account can result in connection errors.

## Distributed configuration for large number of agents

If you plan to use more than 100 connected agents, use a [distributed configuration \(page 1197\)](#). Without a distributed configuration for this situation, you are likely to experience performance issues.

## Sizing recommendations for the Contrast application

The CPU and memory resources for Contrast can vary based on the number of agents connected and application traffic communicating back to the Contrast application. The recommendations on this page apply to the application service.

Additional factors that impact performance include:

- **Web traffic from consumers of Contrast reporting data.**  
Contrast is a highly transactional system that presents calculated and real-time data sets back to consumers of the data. The more users interface with the system, the greater the demand for computing and memory.
- **Large amounts of data maintained in the application over extended periods of time.**  
You can proactively purge data over time or choose to keep the data. With any transactional system, the larger the data set to query against, the greater the computing requirements.
- **More than 100 connected agents**  
If you plan to use more than 100 connected agents, use a [distributed configuration \(page 1197\)](#). Without a distributed configuration for this situation, you are likely to experience performance issues.

Use these guidelines to choose the appropriate mix of resources to scale the requirements to your workload:

- **Small workload:** A small workload is about three to 30 agents communicating to Contrast, and about five to 25 web traffic end users who access the system multiple times a day and actively use alerts, reports and integrations.  
The greater the number of connected agents, the greater the memory requirements are for Contrast to handle in-flight traces. Storage depends on the life of trace data and the preservation of log files by system administrators.

vCPUs	Clock speed	RAM	Storage
~4 to ~8	2.5 GHz to 3.3 GHz	16 GB to 24 GB	100 GB to 200 GB

- **Large workload:** A large workload is about 30 to 100 agents communicating to Contrast, and more than 25 web traffic users for full-scale enterprise deployments. End users access the system multiple times of day, and actively engage in Contrast features such as alerts, reports and integrations. The greater the number of connected agents and end users, the greater the memory requirements for Contrast to handle in-flight traces. Storage depends on the life of trace data and the preservation of log files by system administrators.

vCPUs	Clock speed	RAM	Storage
~8 to ~16	2.5 GHz to 3.3 GHz	24 GB to 48 GB	200 GB to 500 GB



### IMPORTANT

Regardless of your workload size, allocate at least 16 GB of RAM to the Contrast application.



### TIP

Follow the large workload guidelines if you are using the Contrast REST API architecture for automation or data extraction purposes and for continuous integration of agents with large automated regression suites.

## Download Contrast software with curl commands

When your license is provisioned, a good way to download the Contrast installer and licenses is to use curl commands. If you're unsure about who holds access for your company, [contact Support](#).

You can also download other software files using curl commands.

Alternatively, you can download installers and license files from the [Contrast Hub](#).



### NOTE

Starting with Contrast 3.9.10, the installer doesn't include the library and CVE data. For an air-gapped installation, you need to download this data manually. If you have internet access, you can configure your system to [upgrade this data automatically \(page 1216\)](#).

## Steps for downloading installation files

1. To download the latest version of the Linux Contrast installer, use this command:

```
curl -L https://hub.contrastsecurity.com/h/api/artifacts/installer/sh/nocache -u <ContrastHubUsername> -o "contrast-latest-nocache.sh"
```

Replace <ContrastHubUsername> with the username for your Contrast Hub account. After you enter the command, you are prompted to enter the password.

2. To download the latest version of the Windows Contrast installer, use this command:

```
curl -L https://hub.contrastsecurity.com/h/api/artifacts/installer/exe/nocache -u <ContrastHubUsername> -o "contrast-latest-nocache.exe"
```



Replace `<ContrastHubUsername>` with the username for your Contrast Hub account. After you enter the command, you are prompted to enter the password.

3. **Air-gapped installations:** To download the library and CVE data, use this command:

```
curl -JLO https://hub.contrastsecurity.com/h/api/artifacts/ardy_export
-u <ContrastHubUsername>
```

Replace `<ContrastHubUsername>` with the username for your Contrast Hub account. After you enter the command, you are prompted to enter the password.

4. To download the latest Contrast WAR file, use this command:

```
curl -JLO https://hub.contrastsecurity.com/h/api/artifacts/war -u
<ContrastHubUsername>
```

Replace `<ContrastHubUsername>` with the username for your Contrast Hub account.

The WAR file is useful if you have an existing Tomcat server and you want to install Contrast on that server.

5. To download the license file, use this command:

```
curl --request GET --url https://hub.contrastsecurity.com/h/rest/
customer/license/text -u <ContrastHubUsername> > license.lic
```

Replace `<ContrastHubUsername>` with the username for your Contrast Hub account. After you enter the command, you are prompted to enter the password.

6. After you download the files, [install Contrast \(page 1190\)](#).

## Additional software downloads

You can use this curl command to download additional Contrast software:

```
curl -JLO https://hub.contrastsecurity.com/h/api/artifacts/<OtherSoftware>
-u <ContrastHubUsername>
```

Replace `<Other Software>` with any of the these values:

- For the .NET Framework agent, use `dotnet`
- For the .NET Core agent, use `dotnet_core`
- For the .NET Core installer for IIS, use `dotnetcore_installer_for_iis`

## Download the Contrast installer

When your license is provisioned, you get access to a [Contrast Hub](#) account where you can download installers and licenses. If you're unsure about who holds access for your company, [contact Support](#).

Alternatively, you can use [curl commands \(page 1188\)](#) instead of the Contrast Hub to download installers and licenses.



### NOTE

Starting with Contrast 3.9.10, the installer doesn't include the library and CVE data. For an air-gapped installation, you need to download this data manually. If you have internet access, you can configure your system to [upgrade this data automatically \(page 1216\)](#).

## Steps

1. Log in to the Contrast Hub with the credentials provided to you.
2. Download the Contrast installer for your operating system.
  - a. Select **Downloads** in the header.
  - b. Select **Installers**
  - c. Select the file to download:



### TeamServer

Installers					
Version	OS	Release Date	File Name	File Size	
3.8.11.1683721903	linux	01/11/2022	Contrast-3.8.11.1683721903-NO-CACHE.sh	869.91 MB	<a href="#">Download</a> <a href="#">MD5 Sum</a>
3.8.11.1683721903	windows	01/11/2022	Contrast-3.8.11.1683721903--NO-CACHE-x64.exe	874.14 MB	<a href="#">Download</a> <a href="#">MD5 Sum</a>

- **Linux installer:** Contrast<version>-NO-CACHE.sh
  - **Windows installer:** Contrast<version>-NO-CACHE-x64.exe
3. **Air-gapped installations:** [Download the library and CVE data \(page 1215\)](#).  
If you have internet access, you don't have to download the data; you can configure your system to [upgrade library and CVE data automatically \(page 1216\)](#) after installation.
  4. Download the license file.  
The license file is configured with a [SuperAdmin \(page 1269\)](#) account and a regular user account. You'll need the license to complete the installation of the Contrast application.
    - a. Select **Home** in the header.
    - b. Select **Licenses**.
    - c. Select the license file to download.
  5. After you download the files, [install Contrast. \(page 1190\)](#)

## Install Contrast on-premises



### IMPORTANT

These installation instructions are for on-premises use only.

If you are using the hosted version of Contrast, you can [instrument your applications \(page 52\)](#) without installing Contrast. Get started by [installing an agent \(page 52\)](#).

Contrast updates the library data approximately every 24 hours. If you have internet access, your Contrast installation pulls the data from a Contrast database hosted on the cloud. If you don't have internet access (air-gapped installations), you can download the data manually.

## Before you begin

- [Download the Contrast Installer \(page 1189\)](#) from the [Contrast Hub](#) or by using [curl commands. \(page 1188\)](#)

- If you don't have internet access (air-gapped installations), [download the library data manually \(page 1215\)](#).
- Review the [system requirements \(page 1186\)](#) and [sizing recommendations \(page 1187\)](#).

## Steps

1. Preconfigure your base operating system with a shared library for running MySQL. Additionally, the system package `fontconfig` is required on Linux to install fonts. Run the command for your operating system:

- **RHEL 8:**

```
[contrast@myserver ~]# dnf install -y ncurses-compat-libs libaio
fontconfig
```

- **CentOS or RHEL 7:**

```
[contrast@myserver ~]# yum install -y libaio fontconfig
```

- **Ubuntu or Debian:**

```
[contrast@myserver ~]# apt-get install -y libaio1 libaio-dev fontconfig
```


- **Windows:** MySQL requires [Microsoft Visual C++ 2015-2022 Redistributable Update..](#)

2. Run the installer as a privileged user.

- On Windows, right-click on the installer and select **Run As Administrator**.
- On Linux, use the `sudo` command to start the installer.

3. Respond to installer questions according to your situation. (For example, you can [create a MySQL backup \(page 1257\)](#) or [configure the JRE \(page 1203\)](#)).

You can further configure Contrast after startup. You can customize installer behavior using these command line arguments when you run the installation script:

Command line argument	Description
<code>-h -help</code>	Shows help for common command line arguments.
<code>-c</code>	Forces the installation to run in Console mode.
<code>-q</code>	Executes the installer in Unattended mode.
<code>-g</code>	Forces the installation to run in GUI mode. (Windows only).
<code>-console</code>	If the installer is executed in Unattended mode and the <code>-console</code> argument is passed on Windows, a second console shows the output of the installer.
<code>-overwrite</code>	Forces the installer to overwrite all files in Unattended mode regardless of the overwrite policy specified in the installer.
	<div>  <b>CAUTION</b>            This can cause your configuration to be overwritten back to default values.         </div>
<code>-dir</code>	Only valid in Unattended mode; specifies the directory where Contrast should be installed.
<code>-Dinstall4j.debug</code>	By default, the installer catches all exceptions, creates a crash log and informs the user about the location of that log file. This might be inconvenient when debugging an installer; so, this system property switches off the default mechanism, and exceptions are printed to <code>stderr</code> .

Command line argument	Description
<code>-Dinstall4j.keepLog=true</code> <code>-Dinstall4j.alternativeLogfile=[path]</code>	The installer creates a log file prefixed <code>i4j_log</code> for all installations and uninstallation in your temp directory. This log file can be helpful for debugging purposes. If your installer contains an <b>Install files</b> action and terminates successfully, the log file is copied to <code>[installation dir]/install4j/installation.log</code> . Otherwise, the file is deleted after the installer or uninstaller terminates by default.  When using the <code>-Dinstall4j.keepLog=true</code> option, the log file won't be deleted. With the <code>-Dinstall4j.alternativeLogfile=[path]</code> option, the log file is copied to the file specified with <code>[path]</code> . This should be an absolute path name. Neither option has any effect if the log file has already been copied to the installation directory.
<code>-varfile (filename)</code>	Specifies a variable-file to be used. When installing in Unattended mode, this allows you to provide customizations to the default values set by the installer.
<code>--skip-preflight</code>	Skips preflight checks (current user is root, dependencies present). If using this parameter, it must be the first parameter passed on the command line.

**NOTE**

If you're using a distributed setup for the Contrast application, you should use a distributed MySQL instance during setup.

**IMPORTANT**

Client agents use the Contrast URL to communicate back to the application. Contrast makes the best attempt to determine the hostname and pre-populate this value; but, if the provided hostname isn't resolvable by clients on the network, they won't be able to communicate back to the server.

Please set this value to a Contrast host or load balancer that's reachable by your agent hosts.

4. Installation completes and the Contrast application performs its initial configuration. To confirm it has finished, you can visit the URL you specified above.

**NOTE**

If you're [upgrading your version of Contrast \(page 1213\)](#), any required update tasks are included at this point.

5. The first time the Contrast application starts after installation, there are [only two users that can log in \(page 1200\)](#) to the user interface: the default SuperAdmin and the default Organization Administrator. Go to `http://<ContrastServer>:8080/Contrast` (where `<ContrastServer>` is either the IP address or hostname setup during installation), log in as both users and change the password for each.

**IMPORTANT**

To keep your application secure, either disable these default logins and create new ones, or at very least change the default passwords.

6. Once Contrast is installed, the next step is to [configure system settings \(page 1248\)](#) (for example, allocate licenses, set up authentication, and allow users to receive notifications and run reports).

To ensure your library data is current, configure your system to [upgrade library data automatically \(page 1216\)](#). If you don't have internet access (air-gapped installations), [update library data manually \(page 1215\)](#).

## Additional options

After you install Contrast, you have options for expanding your installation:

- Use a [distributed MySQL database \(page 1194\)](#)
- Deploy with a [WAR file \(page 1193\)](#)
- Deploy in a [distributed environment \(page 1197\)](#)

## Deploy Contrast with a WAR file

Using this procedure lets you manage the different components of the Contrast installation separately. After using this method to deploy Contrast, you can upgrade your configuration by replacing the existing WAR file with a new one.

## Before you begin

- [Install Contrast. \(page 1190\)](#)

## Steps

1. Create a compressed file that contains the following files from the directory where you installed Contrast (for example, `/usr/local/contrast`):

- `data/agents/`
- `data/conf/`
- `data/esapi/`
- `data/.contrast`
- `data/.initialized`
- `data/cache/`
- `data/contrast.lic`
- `webapp/Contrast.war`

**Example:** This example shows how to create a TAR file that contains the Contrast files:

```
$ cd /usr/local/contrast
$ tar -czvf ~/ctdc.tar.gz data/agents data/conf data/contrast.lic data/
esapi/ data/cache/ data/.initialized data/.contrast webapp/Contrast.war
```

2. Install Tomcat and Java:

- Use the same version of Tomcat that is included in the Contrast installer.
- [A supported version of Java \(page 1186\)](#)

3. Set up the `contrast-data` directory.

The volume where you create this directory must be large enough for log files, caches and ActiveMQ persistence. For best performance, use a separate volume to handle growth without impacting your overall system.

- a. Create the `contrast-data` directory with a command similar to the following:

```
$ mkdir /opt/contrast-data
```

This example creates the directory in the `/opt` directory, but you can create it in any location.

- b. Unarchive the compressed file you created in step 1 into the `contrast-data` directory.
- c. Check the contents of the directory using a command similar to the following:

```
ls /opt/contrast-data/conf
```

You should see files named `general.properties` and `database.properties` among several others.

- d. To ensure there are no access issues, change the owner and group for the `contrast-data` directory with a command similar to the following:

```
$ chown -R tomcat7:tomcat7 /opt/contrast-data
```

4. To complete the configuration, in the `JAVA_OPTS` environment variable, set the location of `contrast.home` and `contrast.data.dir` to the location where you unarchived the compressed file.

This example shows one way to set the `JAVA_OPTS` variable. Use the documentation for your environment to determine the best way to set this variable.

```
-XX:+UseTLAB
-XX:+UseCompressedOops
-XX:+UseConcMarkSweepGC
-XX:+UseParNewGC
-XX:CMSFullGCsBeforeCompaction=1
-XX:+CMSParallelRemarkEnabled
-XX:+PrintVMOptions
-XX:+PrintCommandLineFlags
-Xmx4g
-Xms4g
-server
-XX:MaxPermSize=768m
-Dcontrast.data.dir=/opt/contrast-data
-Dcontrast.home=/opt/contrast-data
-XX:+HeapDumpOnOutOfMemoryError
-Xloggc:/opt/contrast-data/gc.out
```

5. Place the `Contrast.war` file in the Tomcat `webapps` directory.  
Create a symbolic link, copy, or move the `Contrast.war` file from the location where you unarchived the compressed file to the Tomcat `webapps` directory.
  - This example shows how to create a symbolic link to the file in a Ubuntu environment:

```
$ sudo ln -s /opt/contrast-data/webapp/Contrast.war /var/lib/tomcat7/webapps/Contrast.war
```

- This example shows how to copy the file in a Ubuntu environment:

```
$ cp /opt/contrast-data/webapp/Contrast.war /var/lib/tomcat7/webapps/Contrast.war
```

## Create a distributed MySQL environment

You can use an external MySQL database (an open-source database that runs on both Windows and Linux) with your existing on-premises installation. For example, this is necessary if you are using a [distributed deployment of Contrast \(page 1197\)](#).

**TIP**

Work with your Operations and/or Database team to ensure a secure and durable installation.

You can use a [snippet of Ansible](#) that you can use to install the MySQL on Ubuntu 14.04.

You can also [download the gpg keyfile](#) from MySQL. Contrast changes the bind address to "\*", but recommends binding your MySQL server to the IP of your application server. Create a user and grants that offer access to only the Contrast schema and limited to the host IP address or subnet.

**Steps**

In the following steps, replace `<jdbc.host>`, `<jdbc.port>`, `<jdbc.user>`, `<jdbc.pass>`, and `<jdbc.schema>` with your host, port, user, password, and schema.

1. Install and configure a [supported version \(page 1186\)](#) of MySQL on the database server host.
2. Create a maintenance window for Contrast downtime.
3. [Back up the embedded MySQL database \(page 1257\)](#).
4. Connect to MySQL.

- **Windows:**

```
mysql -h <jdbc.host> -P <jdbc.port> -u <jdbc.user> -p <jdbc.schema>
```

- **Linux:**

```
./mysql -h <jdbc.host> -P <jdbc.port> -u <jdbc.user> -p <jdbc.schema>
```

5. Create the Contrast database with this command:

```
create database <jdbc.schema>;
```

6. Create a MySQL user with this command:

```
CREATE USER '<jdbc.user>'@'%' IDENTIFIED BY '<jdbc.pass>';
```

7. Grant permissions for the Contrast user with this command:

```
GRANT ALL PRIVILEGES ON *.* to '<jdbc.user>'@'%;
```

8. Exit from MySQL.
9. Restore the MySQL backup. Replace `<backup_location>` with your backup location and `<backup_filename>` with your backup filename.

- **Windows:**

```
mysql -h <jdbc.host> -P <jdbc.port> -u <jdbc.user> -p <jdbc.schema> <
<backup_location>/<backup_filename>
```

- **Linux:**

```
./mysql -h <jdbc.host> -P <jdbc.port> -u <jdbc.user> -p <jdbc.schema>
< <backup_location>/<backup_filename>
```

10. Update the configuration in the [encrypted properties editor \(page 1255\)](#). Edit the encrypted file `$CONTRAST_HOME/data/conf/database.properties`. Look for `database.type`; if it doesn't exist, create a new property. Set this value to `distributed` and modify the database connection values to point to the distributed database you want to use.

```
user@ubuntu:/opt/contrast/bin$./edit-properties -e ../data/esapi/
-f ../data/conf/database.properties
```

```

jdbc.type : MYSQL
database.prod.dir : /opt/contrast/data/db
jdbc.debug : false
jdbc.pass : pass
jdbc.schema : contrast
jdbc.host : ubuntu
database.bk.time : 6:39:14
jdbc.port : 3306
database.bk.enabled : false
database.enabled : true
jdbc.url : jdbc:mysql://
ubuntu:3306/contrast
jdbc.user : contrast
database.bk.dir : /opt/contrast/data/
backups/db
jdbc.dialect :
com.aspectsecurity.contrast.teamserver.persistence.CustomMySQL5Dialect
jdbc.driver : com.mysql.jdbc.Driver

Enter the name of the property to edit [q to Quit]: database.type
Create new Property [database.type](y/N): y
Enter a value for the property: distributed

jdbc.type : MYSQL
database.prod.dir : /opt/contrast/data/db
jdbc.debug : false
jdbc.pass : pass
jdbc.schema : contrast
jdbc.host : ubuntu
database.bk.time : 6:39:14
jdbc.port : 3306
database.bk.enabled : false
database.enabled : true
database.type : distributed
jdbc.url : jdbc:mysql://
ubuntu:3306/contrast
jdbc.user : contrast
database.bk.dir : /opt/contrast/data/
backups/db
jdbc.dialect :
com.aspectsecurity.contrast.teamserver.persistence.CustomMySQL5Dialect
jdbc.driver : com.mysql.jdbc.Driver

Enter the name of the property to edit [q to Quit]:

```

**NOTE**

If you're converting from a default embedded database configuration to a distributed configuration, `database.bk.enabled` also needs to be set to `false`. It's your responsibility to configure your own backups when running a distributed database configuration with Contrast.

11. If your on-premises installation is on a Windows system, remove the `contrast-server` service dependency on MySQL.



Before you restart Contrast, remove the `contrast-server` service dependency on the MySQL service with the following command:

```
sc config contrast-server depend= ""
```

## 12. Restart Contrast (page 1201).

### Distributed deployment of Contrast

A distributed configuration of Contrast deploys the database and application server to separate servers. Use a distributed configuration if you:

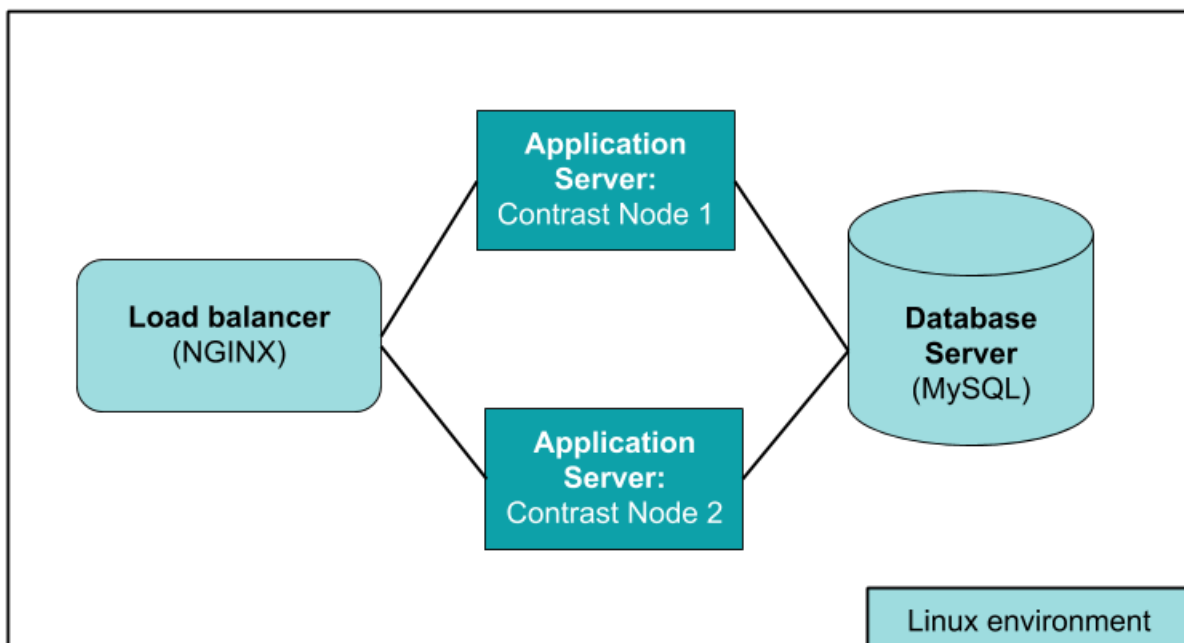
- Plan to use more than 100 connected agents  
Without a distributed configuration for this situation, you are likely to experience performance issues.
- Want to use load-balancing for better performance and scalability
- Require additional administration and management

### Distributed configuration example

This example shows a configuration for installing Contrast in a Linux environment at `/usr/local/contrast`. Your organization may use different environments or have different guidelines on where to install third-party software.

The example shows a configuration using these servers:

- A load-balancer
- A database server
- Two application servers running the Contrast application.  
You can more servers, as needed.



### Before you begin

- Distributed deployment requires an understanding of your environment and the loads it can easily handle.  
To determine whether it's best to use a distributed deployment of Contrast or a dedicated instance, [contact Support](#).
- If you are **already using Contrast**, use your existing instance as **Application Server 1** and be sure it uses a distributed database configuration. Before you continue, you should:

- If you don't already have one, [create a distributed MySQL environment \(page 1194\)](#).
- Determine the version of Contrast running on **Application Server 1** by looking at the contents of the `$CONTRAST_HOME/VERSION` file.
- If the installer you plan to use to create **Application Server 2** is using a higher version of Contrast than **Application Server 1**, you must first [upgrade \(page 1213\)](#) **Application Server 1** to the same version.
- If you are **new to Contrast**, you should:
  - Install and configure MySQL on the database server, as described in steps 1 through 8 in [Create a distributed MySQL environment \(page 1194\)](#).
  - [Install Contrast \(page 1190\)](#) on **Application Server 1** with a distributed database configuration. For example:

```
Choose a MySQL database configuration.
Default [1, Enter], Distributed [2]2
Host
[localhost]
<enter hostname of MySQL server>

Port
[13306]
<enter port to be used to access MySQL server - usually 3306>

Credentials
Username
<enter name of MySQL user that was created for Contrast>

Password
<enter password for MySQL user>
```

## Set up distributed servers

1. Copy the following files from **Application Server 1** to a temporary location on **Application Server 2**:
  - `$CONTRAST_HOME/data/conf/server.properties`
  - The associated Server KeyStore file if you [configured \(page 1203\)](#) **Application Server 1** for HTTPS.
2. If you configured [Single Sign On \(SSO\) \(page 1242\)](#) on **Application Server 1**, complete these steps:
  - a. Run the [encrypted properties editor \(page 1255\)](#) against `$CONTRAST_HOME/data/conf/saml.properties` to retrieve the configured values. Enter `q` at the prompt (you aren't changing any values). For example:

```
$ bin/edit-properties -e data/esapi/ -f data/conf/saml.properties

authenticator.saml.keystore.default.key : some_alias
authenticator.saml.secret.url :
authenticator.saml.keystore.path : /path/to/
samlKeystore.jks :
authenticator.saml.keystore.password : changeit
authenticator.saml.keystore.passwordMap :
some_alias=changeit

Enter the name of the property to edit [q to Quit]: q
```

- b. Create a new file named `saml.properties.cleartext` containing the values you retrieved above, but formatted with an `=` replacing the `:`, for example:

```
authenticator.saml.keystore.default.key=some_alias
authenticator.saml.secret.url=
authenticator.saml.keystore.path=/path/to/samlKeystore.jks
authenticator.saml.keystore.password=changeit
authenticator.saml.keystore.passwordMap=some_alias=changeit
```

- c. Copy the associated SAML KeyStore from **Application Server 1** to a temporary location on **Application Server 2**.
3. [Install Contrast \(page 1190\)](#) on **Application Server 2** with the same distributed configuration that you used for **Application Server 1**.
4. When the installation has completed, [stop \(page 1199\)](#) the Contrast Server on this Application Server.
5. Place the `server.properties` file, the associated Server Keystore, the `saml.properties.cleartext` file and the associated SAML KeyStore (where applicable) in the same directories on **Application Server 2** (usually `$CONTRAST_HOME/data/conf/`).
6. [Start \(page 1199\)](#) the Contrast Server on **Application Server 2**.
7. Test the [default users \(page 1200\)](#) created by the application to be sure they work with both Contrast **Application Servers (1 and 2)**.
8. Set up a load balancer (like NGINX) on the fourth server. If you choose NGINX, use the [basic installation instructions](#).



#### NOTE

Contrast requires sticky or persistent sessions for better performance. For example, with an NGINX load balancer, use the Ip Hash method to guarantee that requests from the same address get to the same server if it's available.

9. Once you set up the server, you must configure Contrast to point to your load balancer. To do this, edit the `/data/conf/general.properties` file on each node. Change the `teamserver.url` value in the YAML config file to that of the load balancer and restart the Contrast application server.

If you are doing health checks for the load balancer, use this URL:

```
<CONTRAST_SERVER>/Contrast/api/public/ng/information
```

where `<CONTRAST_SERVER>` is the host name of the Contrast application server.



#### IMPORTANT

Agents use the Contrast URL to communicate back to the application. Contrast attempts to determine the hostname and pre-populate this value. If clients on the network can't resolve the hostname you provide, they won't communicate back to the server. Please set this value to a Contrast host or load balancer that the agent hosts can reach.

When installation is complete, Contrast begins an initial configuration. It can take two to three minutes to fully start up.

10. To check configuration progress, watch `server.log` and `contrast.log`. When the server successfully starts, you will see something like this in `server.log`:

```
260916 20.18.25,837 {} {} {} INFO (Server.java:303) Contrast TeamServer
Ready -
```

## Run Contrast

To run Contrast:

- **Windows:** In Windows, Contrast is installed as a system service. You can start and stop the service through the Windows Service Manager application.

- **Linux:** The Contrast daemon is registered as an `init.d` daemon. To start and stop the server use:

```
/etc/init.d/contrast-server <start|stop|restart|status>
```

Or

```
service contrast-server <start|stop|restart|status>
```

To start the Contrast server independently of the parent shell, execute:

```
nohup /path/to/installation/contrast/bin/contrast-server start >/dev/null
2>1
```

At this point, it's helpful to tail the server logs:

```
$ tail -f $CONTRAST_HOME/logs/server.log
```

And then the application logs:

```
$ tail -f $CONTRAST_HOME/logs/contrast.log
```

If Contrast starts successfully, you will see this message in the `server.log`:

```
190116 21.22.15,703 {} {} {} INFO (ConnectionTester.java:50) Received code
200 from TeamServer
190116 21.22.15,707 {} {} {} INFO (ConnectionTester.java:60) Server start
has been verified
190116 21.22.15,709 {} {} {} INFO (Server.java:319) Contrast TeamServer
Ready - Took 208323ms
```

## Default and SuperAdmin credentials

As the system administrator who installs Contrast, you can manage the following sets of credentials:

- **Contrast Hub credentials:** New customers receive an email with the username and a link to set the password. You will need these credentials to [download the installer \(page 1190\)](#) and to log in to [Contrast Hub](#).
  - **Username:** Contrast provides the username in the format `example@domain.com`. It is the same username as the Default Organization Administrator.
  - **Password:** Create this password when you select the link in the activation email.
- **Default SuperAdmin credentials:** These credentials are included in the license. They are used for managing the Contrast application in the [role of SuperAdmin \(page 1269\)](#).
  - **Username:** Contrast provides the username in the format `contrast_superadmin@domain.com`, where `domain` is the name of your company's email domain.
  - **Password:** The default password is: `default1!`.
- **Default Organization Administrator credentials:** The Organization Administrator can use these credentials to log in to Contrast after [installation \(page 1190\)](#) and set up and maintain the organization.
  - **Username:** Contrast provides the username in the format `example@domain.com`. It is the same username as the Default Organization Administrator.
  - **Password:** The default password is: `default1!`.



### IMPORTANT

Be sure to change the supplied default passwords as soon as you have successfully logged in. You can reset the SuperAdmin password [in Contrast \(page 597\)](#) or by using command line on [Windows \(page 1248\)](#) or [Linux \(page 1247\)](#).

## Restart Contrast

To restart Contrast:

1. Use the following command(s):

- **Windows:**

```
net stop "Contrast Server"
```

Once the service is completely shut down:

```
net start "Contrast Server"
```

- **Linux:**

```
sudo service contrast-server restart
```

2. At this point, it's helpful to tail the server logs:

```
$ tail -f $CONTRAST_HOME/logs/server.log
```

3. And then the application logs:

```
$ tail -f $CONTRAST_HOME/data/logs/contrast.log
```

4. If Contrast starts successfully, you will see this message in the server.log:

```
190116 21.22.15,703 {} {} {} INFO (ConnectionTester.java:50) Received
code 200 from TeamServer
190116 21.22.15,707 {} {} {} INFO (ConnectionTester.java:60) Server
start has been verified
190116 21.22.15,709 {} {} {} INFO (Server.java:319) Contrast TeamServer
Ready - Took 208323ms
```

## Uninstall Contrast

Each installation comes with a script for safely uninstalling Contrast plus all embedded components such as Java, Tomcat and MySQL. The script is packaged within the root directory of the Contrast installation. On Unix, the file is an executable script labeled *uninstall*. On Windows, a command file is packaged in the installation directory called *uninstall.cmd*.

Before uninstalling:

- [Create a backup of MySQL \(page 1257\)](#) using the database backup tool provided with Contrast.
- Shut down Contrast using either the Windows or Unix service script.

To remove Contrast from your servers using **Windows**:

1. Open the Windows Explorer.
2. Navigate to the Contrast installation directory.
3. Click on the file *uninstall.exe* and run. Run the uninstall with the same privileges you ran the installation (as an administrator).
4. Follow the prompts to perform uninstallation.

To remove Contrast from your servers using **Unix**:

1. Open a Linux console.
2. Change directory (*cd*) to the Contrast installation directory.
3. Execute the command *uninstall*.
4. Follow the prompts to perform uninstallation.

**NOTE**

You'll delete the vast majority of files when performing an uninstallation. However, an administrator may need to delete a few remaining files manually, such as:

- The *Contrast HOME* directory
- The *Contrast DATA* directory
- The *Contrast LOGS* directory
- The *Contrast MYSQL* directory

## Post-installation

After you install Contrast, you have options that might help you improve your Contrast environment.

### Post-installation tasks

Consider these post-installation options:

- [Configure Tomcat \(page 1202\)](#)
- [Configure the Java Runtime Environment \(page 1203\)](#)
- [Configure HTTPS \(page 1203\)](#)
- [Configure HTTP headers \(page 1206\)](#)
- [Customize MySQL \(page 1207\)](#)
- [Configure reporting storage \(page 1209\)](#)
- [Configure logs \(page 1211\)](#)
- [Use Redis as a shared cache \(page 1212\)](#)

### Configure Tomcat

During [installation \(page 1190\)](#), you set some values for the memory used by the embedded Tomcat server on which Contrast runs.

As you add more applications or find more vulnerabilities, you may notice a degradation in performance which can indicate you have reached the maximum amount of memory allowed for this server.

To increase memory settings:

1. Stop the Contrast server by running the `contrast-server stop` command.
2. When the server is stopped and *Contrast/logs/contrast-stdout.log* ends with `[MysqldResource] shutdown complete`, it is safe to change memory settings.
3. In the Contrast bin directory, *c:/Program Files/Contrast/bin* or the default */opt/Contrast/bin*, open a file named *contrast-server.vmoptions*. It should look something like this:

```
-XX:+UseG1GC
-XX:+UseStringDeduplication
-XX:+PrintVMOptions
-XX:+PrintCommandLineFlags
-XX:+UseContainerSupport
-XX:InitialRAMPercentage=50.0
-XX:MaxRAMPercentage=50.0
-XX:MinRAMPercentage=50.0
-Dcontrast.data.dir=/opt/contrast-data
-Dcontrast.home=/opt/contrast-data
-XX:+HeapDumpOnOutOfMemoryError
```

```
-Xloggc:/opt/contrast-data/gc.out
-server
```

4. You can update the values for:

- **Xms**: the amount of memory allocated to the server on start
- **Xmx**: the maximum amount of memory the server can use
- **MaxPermSize**

These values can change depending on the memory available on the machine hosting Contrast.

5. Save the file and start Contrast back up using the `contrast-server start` command.



#### TIP

See [JVM documentation](#) for help with tuning.

## Configure the Java Runtime Environment (JRE)

During [installation \(page 1190\)](#), you are given the option to use either an embedded JRE or a pre-installed JRE.

The minimum supported version is Java 17.

To configure the JRE:

1. Open `$CONTRAST_HOME/install4j/pref_jre.cfg` with a text editor.
2. Add the complete path to the Java version you would like to use. For example:

```
C:\Program Files\Java\jre17
```

## Configure HTTPS

By default, HTTP is used for connections between Contrast and the agents. You may need to add or replace HTTP with HTTPS for both Contrast and agent traffic, which you can do with Tomcat's built-in connector functionality. There are two ways to do this:

- **Contrast HTTPS connector:** Configure Contrast to listen to HTTPS connections on a port that you specify by adding a certificate to a Java KeyStore.
- **Reverse proxy method:** Use a standard web server, such as Apache HTTPD or NGINX, in front of the Contrast server configured to reverse proxy requests using Contrast's AJP connector.

You can customize the configuration further as described in [How-To Modify Supported TLS Versions and Ciphers on On-Premise Contrast Server](#).



#### IMPORTANT

In the following procedures, it is important that you use only a single password throughout. If any of the CA-provided files are password protected, you must either remove that password (your CA can help you with this) or ensure that you use the same password for the resulting JKS KeyStore file.

## Use the Contrast HTTPS connector

Use these procedures to create a Java KeyStore (JKS), with a signed certificate, that your on-premises Contrast application server will use at runtime.



### NOTE

It is also possible to use the HTTPS connector with a self-signed certificate.

## Certificate Signing Request (CSR) is required

For this situation, create the KeyStore first and then use that as the basis for the CSR.



### IMPORTANT

The CA provides you with files that you must import into the same KeyStore from which you generated the CSR.

1. Use the Java keytool command to create a Java KeyStore (JKS) (for example, `contrast.jks`) containing a private and public key for a certificate with an alias of `contrast-server`.

```
keytool -genkeypair -alias contrast-server -keyalg RSA -keystore contrast.jks
```



### NOTE

When you create the KeyStore, depending on the Java version you're using, the first prompt might ask "What is your first and last name?". Enter the Common Name (the FQDN for which the certificate will be issued). For example, use a name like `mydomain.com` instead of your first and last name.

2. Generate a Certificate Signing Request (CSR) (`contrast.csr`). You can add DNS or IP fields as needed to include these as Subject Alternative Names on the certificate.

```
keytool -certreq -alias contrast-server -file contrast.csr -keystore contrast.jks -ext san=dns:your_hostname.your_company.com,ip:10.0.0.1
```

3. Send the resulting CSR file to your CA. The CA will provide you with either multiple PEM files or a single PCKS #7 file.
4. Import the files into the Java KeyStore. Use these instructions depending on the file type you receive.
  - **Multiple PEM files:** These files have extensions of `.CRT` or `.PEM` (PEM files open as readable text). One file contains the certificate, while the others contain the root and possibly one or more intermediate certificates.

The certificates must be imported into the KeyStore in a top-down order, with the server certificate itself being imported last. The server certificate should have the same alias used when the KeyStore was created. For example, if you were provided with `root.cer`, `inter.cer` and `server.cer`, you should import them as:



```
keytool -import -trustcacerts -alias root -file root.cer -keystore contrast.jks
keytool -import -trustcacerts -alias intermediate -file inter.cer -keystore contrast.jks
keytool -import -trustcacerts -alias contrast-server -file server.cer -keystore contrast.jks
```

- **Single PKCS #7 file:** This file has an extension of .P7B, .CER or possibly .CRT. This file contains the server certificate bundled with all necessary root and intermediate certificates. The server certificate should have the same alias used when the KeyStore was created. For example, for a file `certificate.p7b`, import it as:

```
keytool -import -trustcacerts -alias contrast-server -file certificate.p7b -keystore contrast.jks
```

5. Once KeyStore setup is complete, open the `<YourPath>/data/conf/server.properties` file in your text editor, where `<YourPath>` is the path where Contrast is installed. Replace `<port>`, `<full path to>`, and `<password>` with your port, JKS file path, and password.

```
https.enabled=true
https.port=<port>
https.keystore.file=<full path to>/contrast.jks
https.keystore.pass=<password>
https.keystore.alias=contrast-server
```



### IMPORTANT

If using Windows, the full path to the JKS file must be escaped. For example:

```
https.keystore.file=C:\\Program\\ Files\\Contrast\\data\\
\\conf\\ssl\\contrast-server.jks
```

You may find it useful to set the `http.enabled` and `ajp.enabled` options to `false` to ensure that only connections made over HTTPS are allowed to the Contrast server.

6. Open the `<YourPath>/data/conf/general.properties` file, and change the value of the `teamserver.url` property to reflect your change. Agents must be updated manually the first time after you make this change. Future updates to the agent will be automatic.
7. **Optional:** [Modify supported TLS versions and cipher suites.](#)
8. Restart the Contrast server service, and ensure that it's now listening on the HTTPS port you configured.

### No CSR is required

For this situation, create a new KeyStore from the files the CA provides you. If you have an existing KeyStore, delete or rename it before you create the new one.

1. Use one of these methods to create the KeyStore:
  - If you have `server.crt`, `priv.key` and `inter.crt` files: Convert the files to a PKCS #12 and create a KeyStore with these commands.

```
openssl pkcs12 -export -out cert.pfx -inkey priv.key -in server.crt
-certfile inter.crt -name "contrast-server"
keytool -importkeystore -srckeystore cert.pfx -srcstoretype pkcs12
-destkeystore contrast.jks -deststoretype jks
```

- If you have a PKCS #12 file: Create a KeyStore with this command.

```
keytool -importkeystore -srckeystore cert.pfx -srcstoretype pkcs12
-destkeystore contrast.jks -srcalias <sourcealias> -destalias contrast-
server -deststoretype jks
```

2. Once KeyStore setup is complete, open the <YourPath>/data/conf/server.properties file in your text editor, where <YourPath> is the path where Contrast is installed. Replace <port>, <full path to>, and <password> with your port, JKS file path, and password.

```
https.enabled=true
https.port=<port>
https.keystore.file=<full path to>/contrast.jks
https.keystore.pass=<password>
https.keystore.alias=contrast-server
```

3. Open the <YourPath>/data/conf/general.properties file, and change the value of the teamserver.url property to reflect your change. Agents must be updated manually the first time after you make this change. Future updates to the agent will be automatic.
4. **Optional:** [Modify supported TLS versions and cipher suites](#).
5. Restart the Contrast server service, and ensure that it's now listening on the HTTPS port you configured.

## Use the reverse proxy method

To use Apache JServ Protocol (AJP) with the reverse proxy method:

1. Ensure that the Contrast server is configured to listen for connections using the AJP protocol. Open the CONTRAST\_HOME/data/conf/server.properties file in your text editor and verify that the following options are set:

```
ajp.enabled=true
ajp.port=8009
ajp.secretRequired=true|false
ajp.secret=somesecret
```

Choose the `ajp.port` setting to reflect the port on which you'd like the server to listen for incoming connections. If you want the AJP connector to be the only way to access the server, disable the `http.enabled` and `https.enabled` options.

If the `secretRequired` is configured to `true`, the `ajp.secret` setting should have a non-null, non-zero length value. Request workers are required to have the secret keyword; otherwise, the requests are rejected. The workers must provide a matching value, or the request will be rejected regardless of the setting of `secretRequired`.

2. After updating the `server.properties` file, restart the Contrast server service for the changes to take effect.
3. To configure the front-end server, refer to your server's documentation for instructions on how to configure it to use AJP. (For example, see [Apache](#) or [NGINX](#) AJP documentation.)

## Configure HTTP headers

Use this procedure to configure HTTP headers if you are using the Tomcat software that Contrast provides.

When an HTTP header is enabled, it controls whether a document from an HTTP response can be loaded inside a navigable child (for example, `<iframe>`).

The `'X-Frame-Options'` header in the HTML standard provides details about HTTP header configuration.

## Steps

1. Open the `<YourPath>/data/conf/server.properties` file in your text editor, where `<YourPath>` is the path where Contrast is installed..
2. Specify one of these values for the `servlet.response.xframe.options` property:
  - `SAMEORIGIN`: Same-origin embedding is allowed.
  - `DENY`: Embedding is disallowed.
  - No value: The header is omitted. Embedding is allowed.



### NOTE

If the `servlet.response.xframe.options` property is missing, a default value of `SAMEORIGIN` is used.

3. After you update the `server.properties` file, restart the Contrast server service for the changes to take effect.

## Customize MySQL



### NOTE


This procedure applies to the embedded MySQL database, not a distributed configuration.

To match the needs of your environment, you might need to change the default settings that Contrast specifies for the MySQL database. For example, after an upgrade, you might need to adjust the value for the `innodb_buffer_pool_size` setting.'

To change the default MySQL settings, use the `my_extra.cnf` options file. The Contrast installer includes this file as part of the on-premises installation.

Contrast loads its default options file first and then, applies custom settings that you specify in the `my_extra.cnf` file.

## Steps

1. Find the `$CONTRAST_HOME/data/conf/my_extra.cnf` file.
2. Add or change  [MySQL settings](#), as needed.
3. [Restart Contrast. \(page 1199\)](#) or, if you're using Microsoft Windows, restart the MySQL service.

## Set up a proxy configuration

Setting up a proxy configuration for your on-premises installation lets you integrate Contrast with tools that exist inside your network (and don't require a proxy) and outside of your network (which require web proxy access).

### Set up methods

You can use either of these methods to set up a proxy configuration:

- JVM arguments in the `contrast-server.voptions` file.

- The proxy settings in the Contrast Web interface.  
JVM arguments in the options file override settings in the Contrast web interface.

## Steps

- **Use JVM arguments (preferred method)**

1. In the Contrast bin directory, `c:\Program Files/Contrast/bin` or the default directory `/opt/Contrast/bin`, open `contrast-server.voptions`.
2. Add these JVM arguments to the file (for security purposes, `https` arguments are preferred):
  - `https.proxyHost` or `http.proxyHost`  
The host name of the proxy server. Use this argument for communication with externally-hosted software.
  - `https.proxyPort` or `http.proxyPort`  
The port number on which the proxy server listens for traffic.
  - `http.nonProxyHosts`  
A list of hosts, separated by a vertical bar (`|`), which you connect to directly, bypassing the proxy server. This argument works for both `https` and `http` configurations.  
Use this argument to exclude on-premise hosts that are deployed in the same network as your Contrast installation.  
For example, if you have an on-premises Jira installation as well as an integration for MS Teams in the cloud, you can use the `http.nonProxyHosts` argument to exclude the Jira integration from using the proxy server, as shown in the following example:

```
-Dhttps.proxyHost=89.148.22.17
-Dhttps.proxyPort=3128
-Dhttp.nonProxyHosts=jira.mycompany.com|*.internal.mycompany.com
```

In this example, the proxy host is 89.148.22.17, listening for traffic on port 3128.  
This configuration bypasses the proxy server when communicating with the host for `jira.mycompany.com` as well as hosts matching `*.internal.mycompany.com`.

- **Use the Contrast web interface:**

1. In the user menu, select **SuperAdmin**.
2. Select **System settings**.
3. Under Internet settings, select **Proxy**.



4. Specify the proxy host name, the port number, the username for the proxy server, and the password for the proxy server.

5. Select **Save**.

## Configure reporting storage for the system

As SuperAdmin, you can configure reporting storage options by adding the following properties to the `contrast/data/conf/general.properties` file:

- **reporting.storage.mode:** The default value option is `DB`.  
The available value options are `DB` and `FILE_SYS`.  
If you don't specify a value, Contrast uses the default value of `DB`.
- **reporting.storage.path:** This is required when storage mode is set to `FILE_SYS`.

You can also add the same properties to the `contrast-server.vmoptions` file:

- **-Dreporting.storage.mode**
- **-Dreporting.storage.path**

The recommended setting for `reporting.storage.mode` is `FILE_SYS`. When `DB` is configured, files are stored in the database, adding unnecessary contention on the database.

With the `FILE_SYS` option, you must set up a file-sharing service where all Contrast nodes are able to access the file path. Provide this path as the value for `reporting.storage.path`.

**NOTE**

The path should be an absolute path, such as `/Users/user1/reporting`.

For Windows, be sure to escape the colon or the path will not work. For example, this path will fail:

```
reporting.storage.path=C:\Contrast\data\reports
```

You must use either forward slashes or two backslashes around the colon for it to work, like this:

```
reporting.storage.path=C\\:\\Contrast\\data\\reports
```

If an Attestation report exceeds the [report limits \(page 1044\)](#), an error message displays and the report doesn't generate. If this situation occurs, change the report selections when you generate the report to reduce the amount of information in the report.

## Contrast logs

For the Contrast application, Log4j version 2 is used as the logging framework.

You can [configure logging thresholds \(page 1211\)](#), control log file destinations, and see an overview of each log available in Contrast.

Find these logs under the `$CONTRAST_HOME/logs` directory:

- `server.log`
- `catalina.out`

Other logs, like these, can be found in `$CONTRAST_HOME/data/logs`:

- `contrast.log`
- `ldap_ad.log`
- `migration.log`
- `audit.log`
- `mysql_error.log`

This table shows all of the primary log files in Contrast.

Log file	Description
<code>audit.log</code>	Logs audit events such as: <ul style="list-style-type: none"><li>• Logging in and out of the application</li><li>• Impersonating another user</li><li>• Switching organizations</li><li>• Accessing the administrator portal</li><li>• Changes to the configuration and settings of Contrast by a SuperAdmin account</li><li>• User account service issues (locked accounts, password changed, etc.)</li><li>• Deleting traces</li><li>• Changes to a license or an expired license notification</li><li>• API Key changes</li></ul>
<code>console.log</code>	Default application event log
<code>contrast-error.log</code>	Logs messages printed to <code>stderr</code>
<code>contrast-stdout.log</code>	Logs messages printed to <code>stdout</code>

Log file	Description
<i>contrast.log</i>	Like Tomcat's stdout or console log, <i>contrast.log</i> shows most key events happening inside Contrast to inform or help with debugging. It includes information about: <ul style="list-style-type: none"><li>• applications</li><li>• servers</li><li>• libraries</li><li>• traces</li><li>• users</li><li>• Java stack traces for debugging purposes when a server exception takes place</li></ul>
<i>security.log</i>	Formerly the <i>esapi.log</i> , this log file is used for capturing key events from Contrast, such as the loading of a given property file.
<i>migration.log</i>	Contains a summary of all database migrations that occur against the Contrast application between updates. It references the Contrast version, the migration script that ran and the status of the script.

## Configure logs at a system level

Contrast collects [multiple log files that log events and messages \(page 1210\)](#).

You can configure the *log4j2.xml* file used to host the Log4j configuration that is packaged in the Contrast application (and optionally apply [Log4j custom levels](#)).



### CAUTION

before making any changes to this file to ensure the formatting is syntactically correct. A server restart is not required *if the changes are entered correctly*.

1. Find the file under `$CONTRAST_HOME/data/conf`.
2. The first parameter of the file below is a monitor interval that refreshes the settings based on the variable defined. By default, Contrast checks every 60 seconds and refreshes the logging configuration.

```
<Configuration monitorInterval="60">
```

3. Edit the file as needed.

**TIP**

Read more about appenders and delivering LogEvents in the [Log4j documentation](#). Contrast predominantly makes use of the [Rolling File Appender](#), which is an *OutputStreamAppender* that writes to the file named in the *fileName* parameter, and rolls the file over according to the *TriggeringPolicy* and the *RolloverPolicy*.

Here is a sample file of the appenders for *contrast.log*. It shows a daily appender with a 1 GB max file size policy and no more than 15 files of rollover. This appender also compresses the file and renames it daily.

```
<RollingFile name="DAILY" fileName="${contrast.logs.dir}/
logs/contrast.log"
 filePattern="${contrast.logs.dir}/logs/
contrast.%d.%i.log.gz" immediateFlush="true">
 <PatternLayout>
 <Pattern>%d{ddMMyy HH:mm:ss,SSS} {%X{session.id}}
{%X{user.name}} {%X{remote.addr}} %-5p (%F:%L) %m%n
 </Pattern>
 </PatternLayout>
 <Policies>
 <TimeBasedTriggeringPolicy/>
 <SizeBasedTriggeringPolicy size="1 GB"/>
 </Policies>
 <DefaultRolloverStrategy max="15"/>
</RollingFile>
```

The logger section of the file defines which Java packages should log to a specific appender and at a given log level.

## Use Redis as a shared cache (on-premises)

You can configure Contrast to use a shared cache on a Redis server.

This topic does not provide details about setting up Redis servers.

## Contrast properties for Redis

Use these properties:

Property name	Description
cache.userredis	A Boolean value indicating whether Contrast uses Redis for caches.
contrast.cache.redis.db.index	An Integer specifying the database index where Contrast stores cache information.
contrast.cache.redis.proto	A string that controls which protocol to use when Contrast connects to the Redis server.
contrast.cache.redis.host	A string containing the host name or IP address of the Redis server.
contrast.cache.redis.port	An Integer representing the TCP/IP port that the Redis server uses using to listen for new client connections.
contrast.cache.redis.password	A string containing the password used to authorize client access to the Redis server.
contrast.cache.redis.client.name	A string that identifies the client.

## Before you begin

- You need this information:



- The host name or IP address for the Redis server
- The listening port for the Redis server
- The password for a user account on the Redis server
- The database index that the cache should target
- Verify that the Redis server is configured to use TLS (REDISS)

## Steps

1. Create a `contrast.properties` file in the `/data/conf/` folder.  
Ensure this file has the correct permissions to let a `contrast_server` user access it.
2. In the property file, add the Contrast properties for Redis.  
At a minimum, configure values for the host name, listening port, and password for the Redis server.

### Property file example:

```
cache.userredis=true
contrast.cache.redis.db.index=0
contrast.cache.redis.proto=rediss
contrast.cache.redis.host=contrast-redis-server.company.com
contrast.cache.redis.port=6379
contrast.cache.redis.password=changeme
contrast.cache.redis.client.name=contrast
```

3. [Restart Contrast \(page 1201\)](#).
4. **To verify the configuration** look in the `/data/logs/contrast.log` file.  
This file contains all the information on the current configuration and interactions with Redis. Key terms to look for include:
  - RedissonCache
  - Redisson
  - CacheConfiguration

## System updates and upgrades

Periodically, Contrast Security provides updates and upgrades for on-premises Contrast software.

### Updates and upgrades

Use these procedures:

- [Upgrade Contrast \(page 1213\)](#)
- [Upgrade agents \(page 1214\)](#)
- [Update IP address \(page 1215\)](#)
- [Update library data \(page 1215\)](#)
- [Upgrade library data automatically \(page 1216\)](#)
- [Update license \(page 1217\)](#)

### Upgrade Contrast

Contrast releases patches and upgrades as part of the embedded on-premises installer file, which you can download from the Contrast Hub.

The installer intelligently determines that a previous version of Contrast exists on a given system. You can choose to run the updater portion of the process, or run an installation in a separate location. If a previous installation exists, you must configure a parallel installation to run on separate ports.

### Before you begin

Back up the `cacerts` file in the `$CONTRAST_INSTALLATION/jre/lib/security` directory.

The upgrade process overwrites this file which can result in login issues. These certificates are used for [LDAP integrations \(page 1242\)](#).

## Steps

1. [Create a MySQL backup \(page 1257\)](#) and store the backup file on a separate file system or drive to avoid any issues with restoration. The installer attempts to create a database backup as part of the upgrade process, but to be safe, do this manually. Also, back up all configuration files located at `$CONTRAST_HOME/data/conf`.
2. Be sure the Contrast application is running when you start the installation process. Agents will continue to send vulnerability and library messages during this time. When the application initiates a shutdown on its own, the agents defer sending messages until it can reach the application.
3. The upgrade process is nearly identical to the original process to [install the Contrast application \(page 1190\)](#); however, you will be asked to choose to update an existing installation or perform a new installation. You should choose to update an existing installation.
4. The upgrade will initially perform a database backup. Depending on the size of your database, this process can take a few seconds to a few minutes. During this operation, the application should be accessible to agents and end users.
5. The update then deploys a new file system under the installation directory. This primarily consists of deploying the Contrast.war file directory to the `$CONTRAST_HOME/webapps` directory. The application won't be accessible while the file system is updated.
6. Once a successful file system update is complete, the application starts up. While the application is starting up, you can follow along in the log files (*migration.log* and *contrast.log*, specifically). Log entries are written for both file system and database updates in a sequential manner.



### NOTE

Configuration files and database components aren't updated until the initial startup step.

7. The first indicator of a successful update is that you can access the Contrast application by either logging in to the Contrast web interface or using an API request.



### TIP

In the **user menu** you will see a link to the Release News for the version you are using.

8. Review the contents of the *migration.log* immediately after the upgrade. This log will reveal any issues experienced as part of the update process.



### NOTE

In the minutes after the upgrade, deployed agents might attempt to update to the latest agent version. These agents won't reflect their own update until each has restarted and established contact with the Contrast application.

## Upgrade agents (on-premises)

You can download most agents from public repositories. To download .NET and .NET Core agents, go to the Contrast Hub.

Copy downloaded agents to sub-directories in the `$CONTRAST_HOME/data/agents` directory in the Contrast application. On-premises installations of Contrast are automatically configured to support this directory.

## Steps

1. Download the latest version of an agent from the appropriate repository:
  - **.NET:** Download from the [Contrast Hub](#).
  - **.NET Core:** Download the .NET Core agent or the .NET Core agent installer for IIS from the [Contrast Hub](#)
  - **Java:** Download from the [Maven](#).
  - **Node.js:** Download from [NPM](#).
  - **Python:** Download from the [PyPi](#) .
  - **Ruby:** Download from [RubyGems](#).
  - **Go:** Install with a [direct download \(page 533\)](#).
2. Copy the agents to the appropriate sub-directory for each agent language in the `$CONTRAST_HOME/data/agents` directory.  
For example, if you download a Java agent, copy it to the `$CONTRAST_HOME/data/agents/java` sub-directory.  
You don't need to restart the Contrast application. Agents reload dynamically and become accessible for download.

## Update your IP address

Whether you moved the installation, or had to change the hostname or IP address of your Contrast application, you must also complete these steps.

1. Log in to Contrast as SuperAdmin.
2. In the top right, select **SuperAdmin > System Settings > General Settings**.
3. In the **General** panel, change the **TeamServer URL** to `IP:port/Contrast`.
4. Select **Save**.
5. [Restart Contrast \(page 1201\)](#) to apply the changes.

## Upgrade SCA library data manually

Starting with Contrast version 3.7.4, you can download SCA library data manually from the Contrast Hub. This method is useful in situations where you don't have internet access (air-gapped installations).

## Before you begin

- A Contrast Hub account is required.
- For optimal performance, plan to download the library data on a monthly basis.
- If you have a [distributed deployment \(page 1197\)](#) with multiple servers, use the procedure in this topic for one instance. You can use the same downloaded library data files on multiple installations.



### IMPORTANT

For on-premises customers using MySQL 8, the system variable `local_infile` must be set to **ON** so that Contrast can accept CSV files. For more details, see [Security Considerations for LOAD DATA LOCAL](#).

## Steps

1. Log in to the [Contrast Hub](#).
2. Select **Downloads**.
3. Select **Library data exports** and download the archive version you want.

The screenshot shows the Contrast Hub interface. At the top, there's a navigation bar with 'CONTRAST HUB', 'Home', and 'Downloads' (which is highlighted). Below this, the 'TeamServer' section is visible. Under 'TeamServer', there are three tabs: 'Installers', 'Wars', and 'Library Data Exports' (which is selected). The 'Library Data Exports' tab displays a table with the following data:

Release Date	File Name	File Size	Archive	Download	MD5 Sum
09/14/2021	Contrast-Data-Export-202109141732.zip	2.96 GB	<a href="#">Archive</a>	<a href="#">Download</a>	<a href="#">MD5 Sum</a>
09/13/2021	Contrast-Data-Export-202109131732.zip	2.95 GB	<a href="#">Archive</a>	<a href="#">Download</a>	<a href="#">MD5 Sum</a>
09/12/2021	Contrast-Data-Export-202109121732.zip	2.94 GB	<a href="#">Archive</a>	<a href="#">Download</a>	<a href="#">MD5 Sum</a>

4. Extract the downloaded ZIP file and place the CSV files in the Contrast *data/libraries* directory. For example:  
**Unix:** `/etc/contrast/data/libraries`  
**Windows:** `C:\ProgramData\contrast\data/libraries`  
Some files may be hidden due to their names, so ensure all extracted files are moved to this directory.
5. [Restart Contrast. \(page 1201\)](#)  
When Contrast restarts, the data is imported in the background. The CSV files are deleted from the folder as each file is imported.
6. Check the *data/logs/contrast.log* file for success messages as each script completes. For example, you may see a message like this:

```
Beginning CSV import from 'C:\Program
Files\Contrast\data\libraries\java.csv' into 'artifacts_java' Import
temporary table 'artifacts_java' completed, time: 36.6886968s
```

## Upgrade SCA library data automatically

Starting with Contrast version 3.6.4, you can configure the Contrast application to update Contrast SCA library data automatically.

Contrast updates the library data approximately every 24 hours; newly added CVEs are updated every 30 minutes and will then be included in the 24-hour schedule. Your Contrast installation pulls the data from a Contrast database hosted on the cloud.

## Before you begin

- Configure your firewall to allow access to this URL:  
`https://ardy.contrastsecurity.com/production`

- A SuperAdmin role is required.

## Steps

1. Log in to the Contrast web interface as a SuperAdmin user.
2. From the user menu, select **System settings**.
3. Select **General settings**.
4. Under Internet settings, turn on **Contrast Hub**.

**Proxy**

Connect your internet bound traffic

**Contrast Hub**

Contrast's content server where you can get the latest releases and licenses

**Diagnostics** Last Connected 03/02/2022

Send Contrast health statistics to enable proactive support and resolve support cases faster

## Update your on-premises Contrast license

You may occasionally need a new license file. There are two ways to update this file:

- As a SuperAdmin, you can log in to the application and update the license in Contrast.
- You can replace the license file on the local file system. (If the license is expired, you must use this method.)

To replace the license **in the Contrast web interface**:

1. Log in as SuperAdmin.
2. Select **Licensing** in the left navigation.
3. Click **Update this license** at the bottom of the panel.
4. Enter your **Contrast Hub credentials (page 1200)** to download and apply the latest license from the Contrast Hub.
5. If you don't have access to the Contrast Hub, click **Upload license** and paste your license in the field provided.
6. Select **Update**.
7. **Restart Contrast (page 1201)** to apply the new license changes.

To replace the license **in the Contrast file system**:

1. Obtain a new license from the Contrast Hub, your account manager or the technical support team.
2. Rename the new license file *contrast.new.lic*.
3. Stop the Contrast application service.
  - **Windows:** Use the service control panel
  - **Linux:** Execute `sudo service contrast-server stop` or another appropriate command for the distribution configuration. Execute `ps aux | grep contrast` to verify that all Contrast application processes have stopped, and confirm there are no processes listed. If *mysqld* is still running, it may take a few minutes to terminate on its own after stopping the service. If it doesn't terminate, **contact Support**. **Do not kill the processes.**



### IMPORTANT

Don't move the current *contrast.lic* file. Contrast needs both the old and new license files to upgrade the license.

4. Place the new license file in the `<contrast_home>/data` directory.  
On Linux, confirm that the new license file has the same owner, group and permissions as other files in that directory. (Execute `ls -l` to list the directory contents with permissions and owners.) A backup of the current license called `contrast.lic.bak` will be created in the same directory when the new one is consumed during startup.  
Execute `sudo chown contrast_service:contrast_service contrast.new.lic` to change the owner and group.  
Execute `sudo chmod 644 contrast.new.lic` to change the permissions.
5. Start the Contrast application as normal.
  - **Windows:** Use the service control panel.
  - **Linux:** Execute `sudo service contrast-server start` or another appropriate command for the distribution configuration.
6. The new license takes effect.

To update all instances of the Contrast application, follow the steps for the file system method described above for each application instance that's running.

## Manage system administration

Depending on the size of your organization and how you manage your Contrast installation, you can set up [roles \(page 1269\)](#) to best meet your system administration needs.

For a small organization, a single SuperAdmin can manage all system administration work. If you want to share the responsibilities, you can [designate additional SuperAdmins or ServerAdmins \(page 1224\)](#).

- A **SuperAdmin** is responsible for the system administration of Contrast. This may be assigned to one or more individuals. They have access to the **SuperAdmin** option in the user menu, which allows them to configure organizations, applications, servers, vulnerabilities, users and groups.
- A **ServerAdmin** is identical to a SuperAdmin except without access to users or groups. They have access to the ServerAdmin option in the user menu, which allows them to configure organizations, applications, servers and vulnerabilities.

If you have a separate individual or group of individuals that manages end users and agent licenses, you can [add a system access group \(page 1224\)](#) to designate users as **System Administrators** or **Observers**.

- A **System Administrator** is responsible for maintaining organizations and groups. They have access to the SuperAdmin option in the user menu, which allows them to configure organizations, applications, servers, vulnerabilities, users and groups. They can also impersonate administrators at an organization level.
- A **System Observer** has read-only access to organizations, users, applications, groups and traces. They have read-only access to the **Observer** option in the user menu, which allows them to view organizations, applications, servers, vulnerabilities and users.



### NOTE

If a user is designated as **No Access**, they are blocked from system level access to the designated organization(s).

## Manage multiple organizations

On-premises users deploying in a multi-tenant environment can set up Contrast to support multiple organizations within the same system. During the installation process, a default organization is created. After that, users with SuperAdmin credentials can create additional organizations. To do this:

1. Log in to Contrast as SuperAdmin.
2. In the **user menu** select **SuperAdmin** to view system administration options.
3. Select **Organizations** in the header.
4. Select **Add organization**.
5. Supply valid information for the new organization and designate an Organization Administrator by entering credentials of the user who will fill this role.
6. When a user is granted access to the new organization (either in the above step, or by becoming a member of an [organization access group \(page 1164\)](#)) they can move between organizations by selecting the organization name in the **user menu**.



### IMPORTANT

If you are an Organization Administrator and you want to change settings for a particular organization, you must first switch to that organization in the **user menu**, before selecting **Organization settings**. The active organization will show a green check next to its name in the **user menu**.

## Add/edit an organization

In Contrast, an organization is a group of associated users and applications with a shared business purpose. Contrast uses multi-tenant architecture: each Contrast customer is a tenant, represented as an organization.

### Before you begin

- To create an organization, a [System Administrator \(page 1269\)](#) role is required.
- All organizations require a unique name as well as an [Organization Administrator \(page 1267\)](#) to oversee the organization.

### Steps

- **To add an organization:**

1. In the **user menu**, select **SuperAdmin**.
2. Select **Organizations** in the header, then select **Add organization**.
3. In the Add organization window, specify the details for the organization:

Add Organization

Organization Name

Protect

Enable SCA

License Consumption

☐ Require users to manually apply allocated licenses
 ☒ Automatically apply allocated licenses

☒ Assess (Application licenses)
 ☐ Protect (Server licenses)

Language

English

Date Format

MM/dd/yyyy

Time Format

hh:mm a

Time Zone

(GMT-05:00) Eastern Time

Additional Options

☐ Enable Duplicate Vulnerability Notification
 ☐ Enable Route-Based Auto-Verification
 ☒ Enable Security Standards Report
 ☐ Enable DISA STIG Checklist reporting
 ☐ Allow beta language testing
 ☐ Enable Harmony
 ☐ Enable SAST
 ☐ Enable CloudNative

Agent Diagnostics

Application Library Status

BETA

Admin Email

Admin First Name

Admin Last Name

Require Email Activation

Admin Password

Confirm Admin Password

Cancel

Add

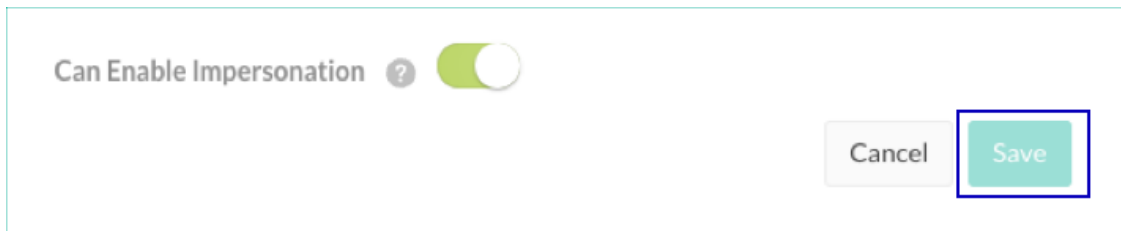
- Enter the new **Organization name**.
  - Use the toggle to enable **Protect**, if appropriate.
  - Use the toggle to **Enable SCA** licensing, if appropriate.
  - Under **License consumption**, use the radio buttons to manually or automatically apply allocated licenses.
  - Select the default **Language** for the organization.
  - Use the dropdowns to choose **Date** and **Time** formats and a **Time zone**.
  - Select additional options for duplicate vulnerability notification, [route-based auto-verification \(page 1126\)](#), [DISA STIG checklist reporting \(page 1045\)](#), [diagnostics \(page 1250\)](#) and any other features.
  - Complete the profile information for the Organization Administrator, including their email, name and password.
  - Only check the box to **Require email activation** if you have a mail server set up with the Contrast application.
4. Select **Add** to create the organization. You may continue to create as many organizations as you need for multi-tenant support.
- **To edit an organization:**
    1. In the **user menu**, select **SuperAdmin**.
    2. Select the name of the organization you want to edit.
    3. Update information as needed and select **Save**. Additional settings in the Edit organization window include:



- **CVSS 3.11:** Contrast is compatible with CVSS 3.1. For on-premises customers, SuperAdmins can enable the scoring by turning on **Enable CVSS 3.1**. Hosted customers must contact [Contrast Support](#) to turn on this setting.



- Hosted customers must contact [Contrast Support](#) to change this setting.  
If you turn this setting off, Organization Admins cannot manage impersonation for their organizations.  
Hosted customers must contact [Contrast Support](#) to change this setting.



## Manage users and permissions at a system level

Before setting up Contrast and adding users, be familiar with Contrast settings for :

- **Users:** You can add users [one at a time \(page 1221\)](#), or [bulk add multiple users \(page 1222\)](#). In the **user menu** select **SuperAdmin**, then select **Users** in the header to see all users and their status (who's awaiting activation, active or inactive, or locked out of their account based on a security policy).
- **Authentication:** You can set up Contrast to use its own internal directory, or use an [external directory like LDAP or Active Directory \(page 1248\)](#).
- **Groups and permissions:** Access and permissions are determined by [roles \(page 1264\)](#). Most roles are assigned with [access groups \(page 1224\)](#). SuperAdmin and ServerAdmin roles are [designated differently \(page 1224\)](#). You can also [grant Protect permissions \(page 1226\)](#) at a system level or when [adding a new user \(page 1221\)](#) or [bulk adding users \(page 1222\)](#).

As a SuperAdmin or System Administrator, you can add users at a system or organization level.

Adding a user to a system group provides them access to the System Administration interface or allows them to perform activities across organizations in cross-organization groups.

You can also add users within a single organization with a defined role to determine their application access and privileges.

### Add or edit a user at a system level

System and Organization Administrators can create users individually, in groups, or through [Microsoft Active Directory \(AD\) \(page 1233\)](#) or [LDAP \(page 1237\)](#) integrations.

### Before you begin

- A System Administrator or Organization Administrator role is required.
- All users are required to have a default organization and a default role within that organization. [SuperAdmin and ServerAdmin roles \(page 1224\)](#) are designated differently.
- When adding an individual user, or multiple users at one time, you can also grant Protect permissions for the users.

## Steps

1. Log in as a SuperAdmin or System Administrator.
2. Select **SuperAdmin** in the **user menu**.
3. Select **Users** in the header.
4. Select a **user name** to edit an existing user or select **Add user** to add a new user.
5. Enter the user's **First name**, **Last name** and **Email address**.
6. Select **Require email activations**, if you want to use email activation instead of requiring a password.
7. Select a **System roles (page 1269)** for the user.  
The default role is **None**.
8. Select the **Organization** to which the user belongs.
9. Select the default **Organization role**.
10. Select a custom or default **Application access group**:  
Contrast provides these default groups:
  - **View**: Members of this group have read-only access to the Contrast interface to see scores, libraries, vulnerabilities, and comments.
  - **Edit**: Members of this group can remediate findings, add tags, manage vulnerabilities, edit attributes, merge applications, add or delete applications, and create servers.
  - **Rules Admin**: Members of this group can edit rules and policies in the application, enable Protect, and manage notifications and scoring.
  - **Admin**: Members of this group can configure and manage settings for an organization.
11. Select a **Date format**, **Time format**, and **Time zone**.
12. To let Organization Administrators change user settings at an organizational level, select **Use organization settings**.  
This option is selected by default.  
To create user settings at a system level, clear the this option.:
  - a. Clear **User organization settings**.
  - b. To restrict users to using the API only and not the Contrast web interface, **Select Make user API only**.
  - c. To let the user see and use Assess data, turn on **Access**.
  - d. To let the user see and use Protect data, turn on **Protect**.



### TIP

You can also [grant Protect permissions \(page 1226\)](#) at an organization level.

13. Select **Add** or **Save**.

## Add multiple users to an organization

You can use a CSV file to add multiple users to an organization.

## Before you begin

- For on-premises customers, a SuperAdmin role is required.
- For hosted customers, an Organization Administrator role is required.

## Steps

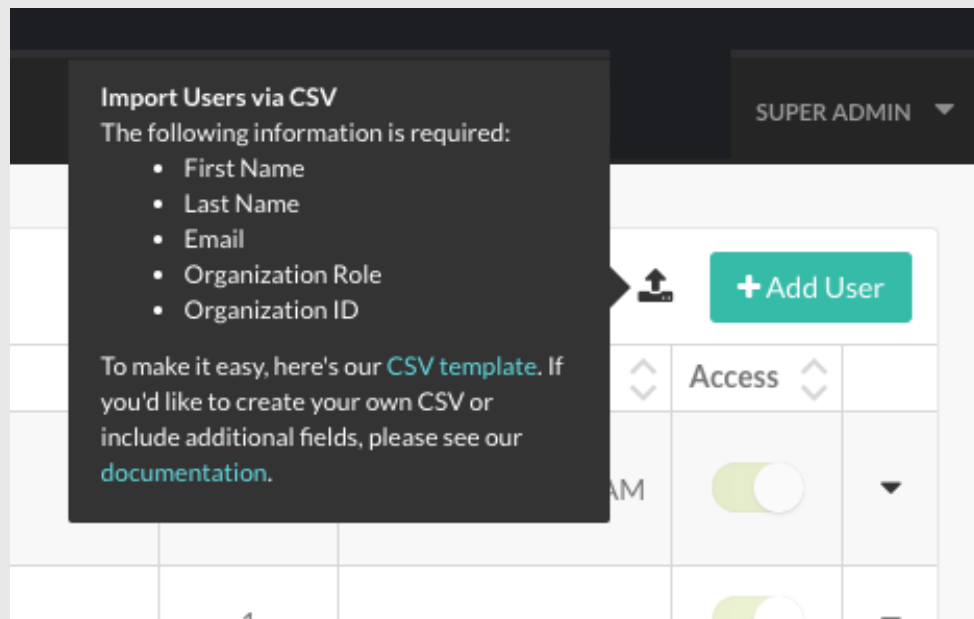
1. Go to the Users list:
  - a. For hosted customers, under the user menu, select **Organization settings** and then, select **Users**.

- b. For on-premises customers, under SuperAdmin, select **Users** in the header.
2. Create a spreadsheet with the recommended information and save it as a CSV file:
  - Include the required fields for each user.
  - Format all field headings and values exactly as shown in the table below.
  - Add a new column for any optional fields.



### TIP

Hover over the **Upload** icon and select the link in the tooltip to download a CSV template to get started.



### CSV fields:

Field name	Required	Value
<b>First Name</b>	Required	User first name
<b>Last Name</b>	Required	User last name
<b>Email or Username</b>	Required	If you are using the Contrast's default internal directory, enter the user's email.  If you are using an external directory, change <b>Email</b> to <b>Username</b> in the CSV, and enter usernames that match those in your external directory exactly.
<b>Organization UUID</b>	Required for on-premises customers	Get this value from the organization's <a href="#">general information settings (page 1160)</a> .
<b>Organization Role</b>	Required	Values can be View, Edit, Rules_admin or Admin.
<b>Date Format</b>	Optional	The default value is the organization setting, such as MM/dd/YYYY.
<b>Time Format</b>	Optional	The default value is the organization setting, such as hh:mm a.
<b>Timezone</b>	Optional	The default value is the organization time zone.
<b>Protect</b>	Optional	The default value is Off.
<b>Groups</b>	Optional	Values can be View, Edit, Rules Admin, Admin or custom group names. Format multiple group names as GroupA&&GroupB&&GroupC.
<b>Language</b>	Optional	The default is the value <a href="#">configured for the organization (page 1160)</a> .
<b>System Administration</b>	Optional	The default value is Off.
<b>Email Activation</b>	Optional	If the value is None, the default is Required Password.
<b>Password</b>	Optional	This field is required if the Email Activation field is set to false.

Field name	Required	Value
Api Only	Optional	The default value is Off.
Access	Optional	The default value is On.

3. Select the black **upload icon** next to **Add user**, then select the CSV you created.  
Once the spreadsheet upload is in progress, you can leave the page and continue with other tasks in Contrast. If the upload is successful, you'll see a confirmation message that includes the number of users uploaded. If the upload failed, you'll see an error message that includes the source of the error in the spreadsheet.

## Designate SuperAdmins or ServerAdmins

The **SuperAdmin** has the highest level system administration permissions.

A **ServerAdmin** has the same permissions and capabilities as a SuperAdmin, except without access to users and groups.

You must have at least one person as SuperAdmin. If you want to designate more than one user as SuperAdmin, do not share a log in, instead:

1. Log in as SuperAdmin.
2. Select **User menu > SuperAdmin > Users**.
3. Find the user you want to designate as SuperAdmin. (You can search by name, email or organization, or find their name in the grid.)
4. Select the user name to open the **Edit user** window.
5. In the **System Administration** field, select **SuperAdmin** or **ServerAdmin**.
6. Select **Save**.



### TIP

Anyone designated as a SuperAdmin or ServerAdmin will be able to access **SuperAdmin** in the top right **User menu**. Also a small key icon will appear next to their name in the **SuperAdmin > Users** grid. Hover over the key to see the assigned role.

## Add, edit or delete a system access group



### NOTE

System access groups are only available to on-premises customers. You must be a SuperAdmin to add a system access group.

To add a system access group:

1. Select **User menu > SuperAdmin > Groups**.
2. Select an existing group to edit, or select **Add group** to create a new group.

**TIP**

To find groups you can use the quick filter dropdown or the search field in the top left, or use the up/down arrows at the top of each column to sort.

3. Fill out the form with:

- **Group name:** Choose something that reflects the purpose, permissions and capabilities you will assign to this group.
- **Type:** Select **System**.

**TIP**

You can also [add an access group at an organization level \(page 1164\)](#). However, if you add access groups at a system level, you have the option of creating cross-organization groups.

Cross-organization groups might be helpful if you have a security team that supports multiple business units that each have their own organization.

Members of a cross-organization group are able to switch between organizations by selecting the name in the user menu.

- **System access:** Select the organization you want this group to access.
  - **Role:** Select the [system role \(page 1269\)](#) you want the members of this group to have within the corresponding organization.
  - Select **Add system access** to add more organizations and roles.
4. Next to **Members**, on the right, type ahead to select one or more users to assign to the group. Select the X to delete members.

**NOTE**

Users can belong to many groups. They don't have to be created within a particular organization in order to gain access to that organization.

5. When you are finished, select **Add** to create the new group, or select **Save** if you are editing an existing group. The members you added to this group will now have permissions that correspond to their role.



## IMPORTANT

If users are assigned to two groups with conflicting roles for all applications or organizations, the role that provides the most restrictive access applies.

Note that only organization and application level groups are [visible to a user \(page 599\)](#), if you are confused about your access level, it may be that stricter permissions have been imposed at a system level.

However, a role assigned to a specific application overrides a role assigned to all applications, even if the application-specific role is more permissive than the role given to all applications.

If a user is assigned to two custom groups that provide roles for the same application, the role with the least privilege applies.

[System \(page 1269\)](#), [organization \(page 1267\)](#) and [application \(page 1264\)](#) roles are listed in order from most to least permissive.

In the following examples of conflicting role permissions, permissions in Group 2 take precedence.

Group 1	Group 2 (takes precedence)
<b>Application Editor</b> for all applications	<b>Application Viewer</b> for all applications
<b>Organization Viewer</b> for all applications	<b>Application Administrator</b> for the Red application
<b>RulesAdmin</b> for the Red application	<b>No Access</b> for the Red application



## TIP

To delete a group, select **User menu > SuperAdmin > Groups**. Find the group you want to delete and select the Delete icon in that row.

Once this is confirmed, the group is removed and any access provided by that group is revoked from all users assigned to the group.

## Grant Protect permissions (on-premises)

For on-premises customers with multiple organizations, you can grant permissions that let all or some user roles in one or more organizations access Protect data.

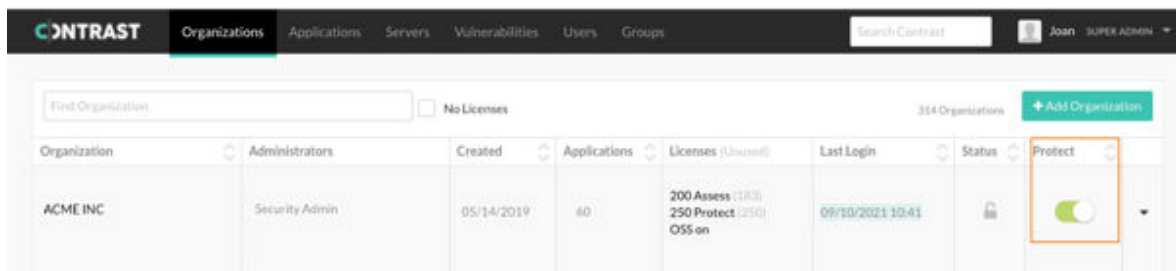
### Before you begin

- A SuperAdmin role is required.
- Identify the organizations whose users need access to Protect data.
- Identify which user roles in an organization need access to Protect data.

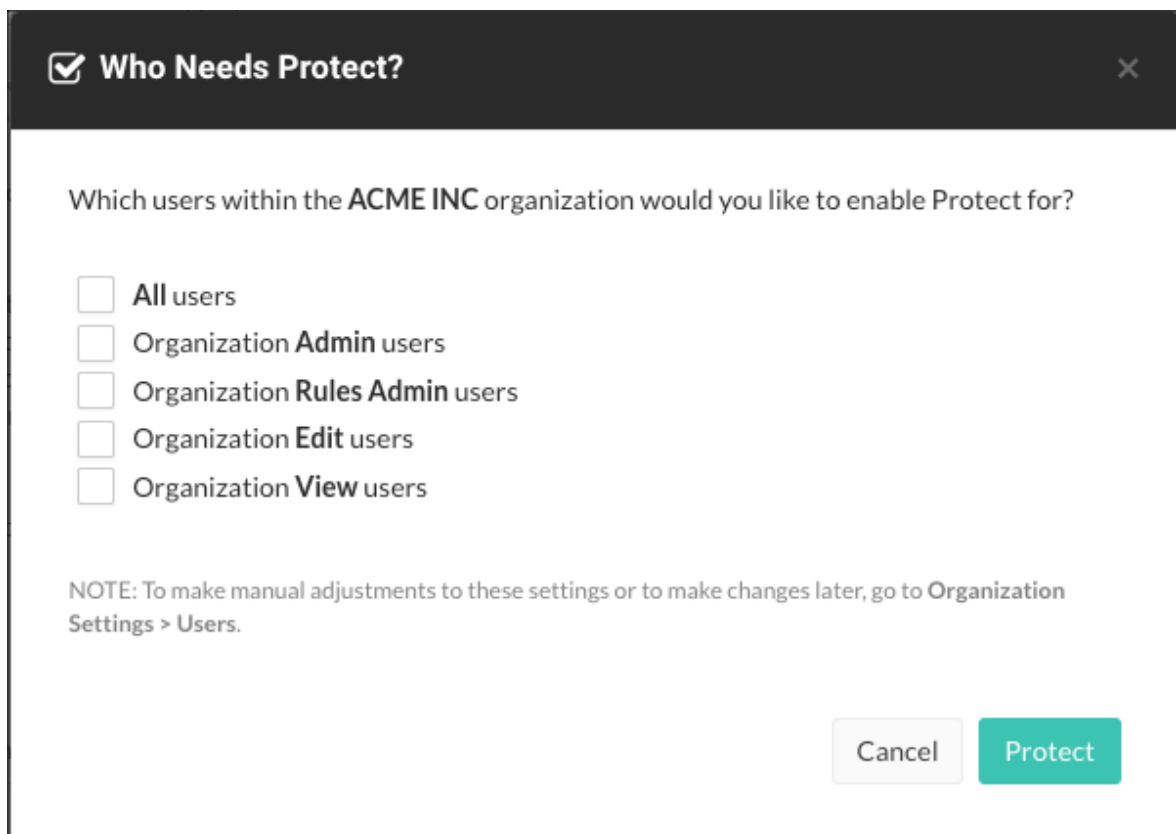
### Steps

1. Log in as SuperAdmin.
2. Select **SuperAdmin** in the user menu.
3. Select **Organizations** in the header.

- Find the organization for which you want to enable Protect. In the Protect column for that row, turn the setting on.



- In the Who needs Protect window, select the roles that need permission to see and access Protect data.



Select **All users** or specific user roles.

You can also enable or disable Protect access for [individual users](#) (page 1221).

- Select **Protect**.  
When you make this change, the users with the selected roles have access to Protect data.

## Automatically add users to groups with SSO

You can automatically add users to groups with single sign-on (SSO).

- Update your SAML configuration in your IDP:

```
<saml2:AttributeStatement
xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">
 <saml2:Attribute Name="contrast_groups"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
 <saml2:AttributeValue xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance"
```

```

 xsi:type="xs:string"
 >GROUP1</saml2:AttributeValue>
 <saml2:AttributeValue xmlns:xs="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance"
 xsi:type="xs:string"
 >GROUP2</saml2:AttributeValue>

 ...
 </saml2:Attribute>
</saml2:AttributeStatement>

```



### IMPORTANT

The attribute values listed under `contrast_groups` must exactly match an existing group name. Contrast won't create new groups based on the values listed under this attribute.

- Then in Contrast, under [organization settings \(page 1158\)](#), select **Single sign-on** and use the check boxes at the bottom of the form to enable one or both of these:
  - Add users to their Contrast groups upon SSO login:** Upon login, Contrast adds users to groups listed in the `contrast_groups` attribute in the SAML assertion.
  - Remove users from their Contrast groups upon SSO login:** Upon login, Contrast removes users from groups not listed in the `contrast_groups` attribute in the SAML assertion.

## References

- User email as NameID

```

<md:NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress</
md:NameIDFormat>

```

- First name and surname

```

<saml2:Attribute Name="http://schemas.xmlsoap.org/ws/2005/05/identity/
claims/givenname"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified"
>
 <saml2:AttributeValue xmlns:xs="http://www.w3.org/2001/
XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance"
 xsi:type="xs:string"
 >Dan</saml2:AttributeValue>
</saml2:Attribute>

<saml2:Attribute Name=" http://schemas.xmlsoap.org/ws/2005/05/identity/
claims/surname"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified"
>
 <saml2:AttributeValue xmlns:xs="http://www.w3.org/2001/
XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance"
 xsi:type="xs:string"

```



```
>Dan</saml2:AttributeValue>
</saml2:Attribute>
```

- User group management

```
<saml2:AttributeStatement
xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">
<saml2:Attribute Name="contrast_groups"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
<saml2:AttributeValue
xmlns:xs="http://www.w3.org/2001/XMLSchema"xmlns:xsi="http://www.w3.org/
2001/XMLSchema-instance"xsi:type="xs:string">GROUP1</saml2:AttributeValue>
<saml2:AttributeValue
xmlns:xs="http://www.w3.org/2001/XMLSchema"xmlns:xsi="http://www.w3.org/
2001/XMLSchema-instance"xsi:type="xs:string">GROUP2</saml2:AttributeValue>
...
</saml2:Attribute></saml2:AttributeStatement>
```

## See also

🔗 [Configuring user and group provisioning with Okta](#)

🔗 [Configuring ADFS to automatically add users to groups](#)

## Default and SuperAdmin credentials

As the system administrator who installs Contrast, you can manage the following sets of credentials:

- **Contrast Hub credentials:** New customers receive an email with the username and a link to set the password. You will need these credentials to [download the installer \(page 1190\)](#) and to log in to [Contrast Hub](#).
  - **Username:** Contrast provides the username in the format *example@domain.com*. It is the same username as the Default Organization Administrator.
  - **Password:** Create this password when you select the link in the activation email.
- **Default SuperAdmin credentials:** These credentials are included in the license. They are used for managing the Contrast application in the [role of SuperAdmin \(page 1269\)](#).
  - **Username:** Contrast provides the username in the format *contrast\_superadmin@domain.com*, where *domain* is the name of your company's email domain.
  - **Password:** The default password is: default1!.
- **Default Organization Administrator credentials:** The Organization Administrator can use these credentials to log in to Contrast after [installation \(page 1190\)](#) and set up and maintain the organization.
  - **Username:** Contrast provides the username in the format *example@domain.com*. It is the same username as the Default Organization Administrator.
  - **Password:** The default password is: default1!.



### IMPORTANT

Be sure to change the supplied default passwords as soon as you have successfully logged in. You can reset the SuperAdmin password [in Contrast \(page 597\)](#) or by using command line on [Windows \(page 1248\)](#) or [Linux \(page 1247\)](#).

## Impersonate users

Impersonating users lets you access organizations as if you have the same role and permissions as the impersonated user. Impersonation is helpful when you need to troubleshoot issues.

If you have a ServerAdmin or System Admin role, you can impersonate the first Organization Admin for an organization by selecting **Impersonate** for an organization on the Organizations page. To use impersonation, you must have access to these organizations.

If you have a ServerAdmin or System Admin role, you can impersonate the first Organization Admin for an organization by selecting **Impersonate** for an organization on the Organizations page. To use impersonation, you must have access to these organizations.

Contrast turns off impersonation automatically after 24 hours.

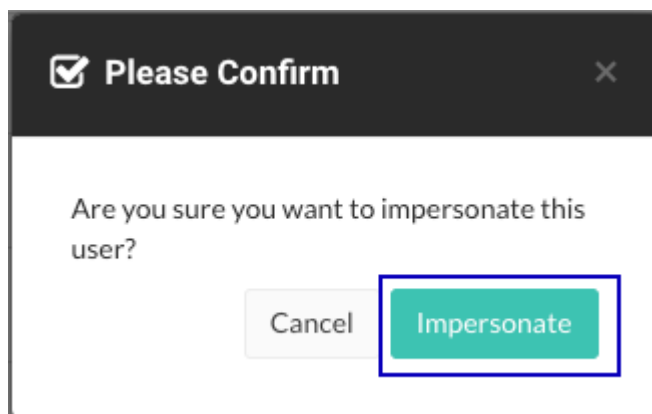
## Before you begin

- The user you choose to impersonate must have an [organization role \(page 1267\)](#).
- If the Impersonation option is not visible for an organization, a SuperAdmin must turn on the [Can enable impersonation \(page 1219\)](#) setting for the organization.
- An Organization Admin must [enable impersonation \(page 1184\)](#) for the organization that you want to access.

## Steps

### • To start impersonation at a user level:

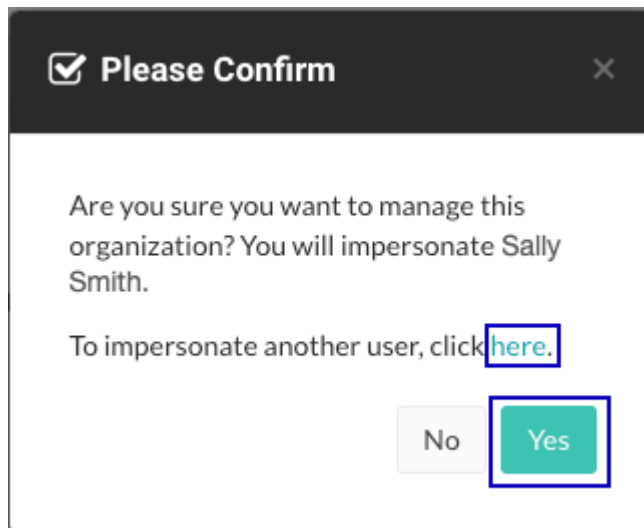
1. Log in to the Contrast web interface as a SuperAdmin.
2. Select **Users**.
3. At the end of the row for the user you want to impersonate, select the triangle ( ▾ ) and select **Impersonate**.
4. In the Please confirm window, select **Impersonate** to confirm that you want to impersonated the selected user.



Contrast opens a session as the impersonated user.

### • To start impersonation at an organization level:

1. Log in to the Contrast web interface as a SuperAdmin, ServerAdmin, or System Administrator.
2. Select **Organizations**, if not already selected.
3. At the end of a row for an organization you want to access, select the triangle ( ▾ ) and select **Impersonate**.
4. In the Please confirm window, select **Yes** to confirm that you want to impersonate the displayed user or select a different user who is an administrator for the organization you want to access.



## Configure authentication



### NOTE

For hosted users, Contrast Security configures authentication. However, an Organization Administrator may be granted the ability to override these settings, including SSO setup.

To request this change contact [Contrast Support](#).

By default, Contrast stores a user directory that includes user's login name, credentials and other details about application authentication. Database credentials are stored (using a one-way hash) in the internal Contrast database. You can set a [password policy \(page 1246\)](#) and [two-step authentication \(page 1232\)](#) for users in Contrast.

Alternatively, you can use an external directory for authentication, in which case, only the username is stored in the Contrast database. Contrast supports:

- [LDAP \(page 1237\)](#)
- [Active Directory \(page 1233\)](#)
- [Single Sign-On \(page 1242\)](#)
- [Trusted HTTPS Proxy \(page 1246\)](#)

Any changes to the authentication settings require that you [restart Contrast \(page 1201\)](#). To change authentication, select **Authentication** under [system settings \(page 1248\)](#).



## IMPORTANT

When you switch between authentication modes, be aware:

- Any users that were created under the previous authentication mode will no longer work, unless the user's email address is the same between the new and old authentication provider.
- After you setup your new authentication mode and restart your server, users can't login to Contrast until their accounts are added to a new or existing organization. For on-premises customers, the SuperAdmin can take care of the Organization Administrator accounts, and then each Organization Administrator is responsible for the users within that organization.



## NOTE

When you use an external authentication provider mode (LDAP or AD), the username field when adding a user functions as a live search that shows users in the proper group.



## TIP

Since roles and permissions are managed by access groups (and not the user directory), it is best practice to create access groups before configuring authentication. You will need at least two unique access groups, one for administrators and one for users.

## Turn on multi-factor authentication at a system level

### Before you begin

- If you configure SSO in Contrast and also want to use multi-factor authentication, configure it using your identity provider (IdP), instead of Contrast. With SSO configured, Contrast passes the responsibility of authenticating users to the IdP.
- For additional protection, organization administrators can set [multi-factor authentication \(page 1168\)](#) as required at the organization level.

### Steps

1. Under [system settings \(page 1248\)](#), select **Security** in the left navigation.
2. Turn the toggle on (green) to enable multi-factor authentication.
3. Select **Allow organization override** to allow Organization Administrators to [choose whether or not to require the feature for users \(page 1168\)](#).

**NOTE**

If a user belongs to multiple organizations, their default organization determines their multi-factor authentication settings.

A user can also [choose how they want to receive multi-factor authentication notices](#). [\(page 597\)](#)

## Configure Microsoft Active Directory

As a [System Administrator \(page 1269\)](#), you can configure Contrast to connect to a Microsoft Active Directory (AD). Use the AD connector to configure this integration. AD has a well-defined structure for directories, resulting in fewer possibilities and a more direct configuration.

**NOTE**

Switching to AD from a different authentication method such as a local database may result in issues if the user ID attribute is inconsistent.

## Access if AD is offline

In the event that your AD service experiences connectivity or configuration issues, use the default SuperAdmin account to log in to Contrast. This feature ensures that you continue to have immediate access to your Contrast account even when AD is offline.

## Steps

1. Start by creating a user in the Active Directory Server that is a dedicated read-only user. The user should have read permission to the directory, including users with permission limited only to the Search Base. You will need this user to set the Search Base when [configuring users for AD \(page 1236\)](#) and when binding to user.
2. Create user groups in the external AD server. You will later use these groups to [assign SuperAdmin privileges \(page 1235\)](#) in Contrast.
3. Under [system settings \(page 1248\)](#), select **Authentication**.
4. Select **Change authentication method** and follow the steps to configure server, groups and advanced settings.
5. Select **Active Directory**.
6. Enter the following information. Some settings may be different for LDAP over SSL (LDAPS) as noted.

#### Under **Connect server**:

- **Protocol:** Enter the protocol that should be used to communicate with the LDAP server. Choose **LDAP** or **LDAPS** from the dropdown. The default is **LDAP**. Additional configuration may be needed for the **LDAPS** option if you are [using a self-signed or privately-signed certificate with AD \(page 1236\)](#).
- **Hostname:** Enter the hostname to connect to when communicating with the LDAP server; either the DNS hostname or IP address of the AD server. In multi-tenant forests, this should be the Global Catalog Server. The default is `localhost`.
- **Port:** Enter the port to connect to when communicating with the LDAP server. For standard (single-tenant, single-domain) directories, this should be port **389 (LDAP)** or **636 (LDAPS)**. In multi-tenant or multi-domain forests, this should be **3268 (LDAP)** or **3269 (LDAPS)**.
- **Search base:** Enter the base DN (a distinguished name that represents the global base level container for your AD environment) to use when communicating with the LDAP server. This is usually your domain or subdomain name. The default is `dc=contrastsecurity,dc=com`. If your login domain is `yourdomain.com`, your base DN would be `dc=yourdomain,dc=com`.

#### Under **Bind to server**.

- **Username:** Enter the full DN of the user who can bind to the directory to perform search functionality. The default is `cn=Directory Manager`.
  - **Password:** The password of the user account that the application should use when connecting to the LDAP server.
7. Select **Test connection** to ensure connectivity to the server. Once connectivity is verified, select **Next**.
  8. [Configure groups. \(page 1235\)](#)
  9. [Configure advanced settings. \(page 1236\)](#)
  10. Once all of the configuration options are set, verify that you are able to log in as both a SuperAdmin and an Organization Administrator using the **Test login** button.



#### NOTE

If the test seems to take an excessive amount of time, it's likely a result of having the wrong setting for the **Follow referrals** option in [Advanced settings \(page 1236\)](#). Once you switch the setting, you should be able to verify login functionality more quickly.

11. Select **Finish** to complete the configuration.

## Configure groups for Active Directory

As part of [Active Directory configuration \(page 1233\)](#), you will need to configure groups.

Contrast doesn't perform Data Access Control using the integrated AD servers. In other words, roles and access to data within the application are handled by the application and Organization Administrators set user roles. However, there is an Access Control check when logging in or creating new users to ensure that the provided user belongs to the correct group in Active Directory (AD).

### Steps

## STEP 3 Configure Groups ?

### Contrast User Group

[Query for Groups](#)

### Contrast SuperAdmin Group

1. Use the groups that you created on your external AD server to assign users to one of the following Contrast groups.
  - **SuperAdmin group:** This group allows users to log in to the Super Administrator interface. Users in this group are authenticated and authorized the first time they log in to Contrast.
  - **User group:** This group allows users to be added to an organization and log in to the Contrast web interface. This group is appropriate for all other users.  
To let users log in, [add them \(page 1163\)](#) to the organization manually in the Contrast web interface.



#### NOTE

If you add a user to both groups in your AD instance, Contrast automatically adds them to the SuperAdmin group during configuration.

2. Select **Query for groups** to enable a live search of existing groups as you begin to type within the input fields.



#### NOTE

To create users with AD authentication in Contrast while bypassing the Access Control check, execute the following query in the database.

```
UPDATE teamserver_preferences
SET property_value='true' WHERE
property_name='directory.skip.user_existence.validation'
```

## Configure settings for Active Directory

As part of [Active Directory configuration \(page 1233\)](#), under **Advanced settings** enter:

**Set Up Authentication**

**STEP 4**  
**Advanced Settings**

User Base DN

User ID Attribute  
☐ Login ID ☒ Email Address ☐ User Principal

Search Within Nested Groups  
☐

Follow Referrals ☒

Limit Referrals

Previous Test Login Finish

- **User base DN:** The default is *cn=Users*, the default container for AD. However, if your AD is configured differently, this will be the path to the top-most container where users are stored in the directory.  
For example, if your users are stored in the DN *CN=Engineering,CN=GlobalUsers,DC=intranet,dc=example,dc=com* and your base DN is *DC=intranet,DC=example,DC=com*, the value of the User DN suffix will be *CN=Engineering,DC=GlobalUsers*.
- **User ID attribute:** Enter the user attribute that the user will enter as the username when logging in to the Contrast application. Use the attribute that will be most familiar to the users. The default is **Email address**.
  - **Login ID:** The AD *sAMAccountName* attribute; this is usually the username that you use to log in to Windows.
  - **Email address:** The AD *mail* attribute containing the email address of the user.
  - **User principal:** The AD *userPrincipalName* attribute containing the full username.
- **Search within nested groups:** Enable or disable searching within nested groups. The toggle is disabled by default.
- **Follow referrals:** In multi-tenant or multi-domain enterprise forests, LDAP queries may be referred to another server for more details. The toggle is disabled by default.
- **Limit referrals:** Limit to how many referrals should be followed when AD replies with a *Referral* response. The default is "5".

## Use self-signed or privately-signed certificates with Active Directory

If you [configure your AD integration \(page 1233\)](#) to connect to your server using SSL, you may need to import your server's certificate into a new truststore to be used by the Contrast JRE.

1. Acquire the server's certificate from your administrators in PKCS#12 format. If you're using a self-signed certificate, this will be the actual AD server's certificate. If you have a private CA, you need the CA certificate for that server.



2. Once you have the certificate for the server, create a truststore that contains that certificate. Run the following commands as an administrator from a command shell in the directory where Contrast is installed.

```
$ mkdir data/conf/ssl
$ jre/bin/keytool -import -file <path to certificate> -alias <hostname> \
 -keystore data/conf/ssl/truststore.jks
```

3. After you create your truststore containing either your server's or CA certificate, add the following lines into the *bin/contrast-server.vmoptions* file:

```
-Djavax.net.ssl.trustStore=<full path to truststore>
-Djavax.net.ssl.trustStorePassword=<password you set for the trustStore,
if any>
```

4. You should now restart the Contrast server service, and verify that queries against AD will use SSL.

## Configure LDAP

Contrast integrates with many types of Lightweight Directory Access Protocol (LDAP) servers. LDAP is an Internet protocol that web applications can use to look up users or groups listed on an LDAP directory server.

Contrast supports these LDAP server types:

- OpenLDAP
- OpenDS
- ApacheDS
- Fedora Directory Server
- Microsoft Active Directory
- Generic LDAP Servers

Connecting to an LDAP directory server is useful if you manage users and groups in a corporate directory, and you want to delegate the responsibility of managing user access of the application to your corporate directory administrators.



### NOTE

Switching to LDAP from a different authentication method such as a local database may result in issues if the user ID attribute is inconsistent.

A system administrator can configure the LDAP server:



### IMPORTANT

If you configure your LDAP integration to connect to your server using SSL, you may need to take extra steps for [self-signed or privately signed certificates \(page 1242\)](#).

## Access if LDAP is offline

In the event that your LDAP service experiences connectivity or configuration issues, use the default SuperAdmin account to log in to Contrast. This feature ensures that you continue to have immediate access to your Contrast account even when LDAP is offline.

## Steps

1. Start by creating a user in the LDAP Server that is a dedicated read-only user or read-write user (depending on how you configure Contrast to interact with the LDAP directory). The user should be have read permission to the directory, including users with permission limited only to the Search Base. You will need this user to set the Search Base when [configuring users for LDAP \(page 1241\)](#) and when binding to user.
2. Create user groups in the external LDAP server. You will later use these groups to [assign SuperAdmin privileges \(page 1239\)](#) in Contrast.
3. Under [system settings \(page 1248\)](#), select **Authentication**.
4. Select **Change authentication method**.
5. Select **LDAP**.
6. Enter required information under **Connect server** and **Bind server**.

Option	Description	Default
<b>Connect server</b>		
<b>Protocol</b>	The protocol that should be used to communicate with the LDAP server. Choose between <b>LDAP</b> or <b>LDAP with SSL (LDAPS)</b> .	<b>LDAP</b>
<b>Hostname</b>	Enter the hostname to use when communicating with the LDAP server.	localhost
<b>Port</b>	Enter the port to use when communicating with the LDAP server.	<b>389 (LDAP), 636 (LDAPS)</b>
<b>Search base</b>	Enter the base distinguished name (DN) to use when communicating with the LDAP server. If your login domain is <i>yourdomain.com</i> , your base DN would be <i>dc=yourdomain,dc=com</i> .	<i>dc=contrastsecurity,dc=com</i>
<b>Bind to server</b>		
<b>Method</b>	Select the method to use when connecting to the LDAP server. Options are shown in the next table.	Simple
<b>Username</b>	Enter the full DN of the user that should bind to the directory to perform queries and check authentication.	N/A

Option	Description	Default
Password	Enter the password for the bind user specified in the <b>Username</b> field.	N/A

There are four supported bind mechanisms that can be used by Contrast. Each has different required fields:

Method	Description	Required Fields	Optional Fields
Anonymous	Administrators provide anonymous, read-only access to the directory.	None	N/A
Simple	This is standard username and password authentication. The username and password are verified as provided by the LDAP server.	Username, Password	N/A
DIGEST-MD5	A username and password are provided and hashed using MD5 prior to sending to the server to be authenticated.	Username, Password	SASL Realm
CRAM-MD5	The LDAP server issues a pre-authentication challenge, which is sent with the MD5 hashed username and password to be authenticated.	Username, Password	SASL Realm

- Once you configure the connection to the LDAP server, select **Test connection**. Testing the connection ensures that the application can connect to the LDAP server and perform queries.
- [Configure groups for LDAP \(page 1239\)](#).
- [Configure users for LDAP \(page 1241\)](#).
- To verify that the group and user mappings are correctly configured, select **Test login**.
- Once you've successfully logged in as both SuperAdmin and Organization Administrator, select **Finish** to complete the configuration.

## Configure groups for LDAP

As part of the [LDAP configuration \(page 1237\)](#), you will need to configure groups.

Organization Administrators set the [roles and permissions \(page 1264\)](#) for users, and each application handles roles and access to data within that application. When configuring users, you can opt to [add users to an access group on login \(page 1241\)](#). However, even if that is not enabled, Contrast uses the LDAP directory to ensure that the provided user belongs to the correct group.

To configure groups:

- Enter the following values:

Option	Description	Default
<b>Group type</b>	Groups types depend on your server functionality and configuration. Groups are either: <ul style="list-style-type: none"> <li><b>Static:</b> Groups track members through an attribute on the object, such as <code>uniqueMember</code>. The remaining four options in this table only apply to static groups.</li> <li><b>Dynamic:</b> The user object tracks its own membership. Groups are added dynamically to the user object when the user becomes a member of a group.</li> </ul>	<b>Static</b>
<b>Group subtree</b>	Configures whether subtrees of the Base DN should be included when searching for groups in the directory.	<b>Enabled</b>
<b>Base DN</b>	This is the distinguished name (DN) where the application can find groups in your LDAP server (like the User Base DN).	<code>ou=Groups</code>
<b>Object class</b>	If left blank, the application uses the default values of "group," "groupOfUsers," or "groupOfUniqueUsers." This isn't a required field, as it is standard across LDAP deployments.	N/A
<b>Group member attribute</b>	The attribute within a group object in the directory that contains the members of that group. This may differ for your LDAP deployment, so ensure that you are using the correct attribute with your LDAP administrator. <p>Each member of the group should be listed as a full distinguished name (DN) not a relative distinguished name (RDN). For example: "cn=smith,ou=Users,cn=support,dc=test,dc=org".</p> <p>If you use an RDN, Contrast does not see that user in the LDAP group.</p>	<code>uniqueMember</code>

- Use the groups you previously created in your external LDAP server, to assign users to one of the following groups
  - SuperAdmin group:** This group allows users to log in with SuperAdmin permissions.
  - Users group:** This group allows users to be added to an organization and log in to the standard interface. This group is appropriate for all other users.



### IMPORTANT

If a user belongs to both groups, and provisioning is disabled, the user will be created as a SuperAdmin. If provisioning is enabled, the user will be created without SuperAdmin permissions.

3. Select **Query for groups** to enable a live search of existing groups as you begin to type within the input fields.

## Configure users for LDAP

As part of the [LDAP configuration \(page 1237\)](#), you will need to configure users.

To fully integrate with an LDAP directory, Contrast needs information on how to connect to the LDAP server as well as how to find users and groups within the directory.

1. Enter the following information on how you want Contrast to search for users in the directory. The default settings are correct for most LDAP deployments.

**System settings**

General Settings  
Licensing  
Policy  
Security  
API  
**Authentication**  
Mail  
Score Settings  
System Messages

**Set Up Authentication**

Step 4 Configure Users

**Find Users**

Base DN  
ou=People,dc=example,dc=com

Object Class  
inetOrgPerson

First Name Attribute  
givenName

Last Name Attribute  
sn

Email Attribute  
mail

User Subtree ☒

User ID Attribute  
mail

Authentication Strategy ☒ Bind ☐ Compare

Default Organization  
New Org

Default Organization Role (New Users)  
Edit

Default Application Access Group  
Edit

☒ Enable user provisioning  
☐ Add users to their Contrast groups upon login  
☐ Remove users from their Contrast groups upon login

Previous Test Login Finish

Option	Description	Default
Base DN	Indicate the container (under the global base DN) where Contrast should start searching for users. In most organizations, this is a single container (for example, <code>ou=Users</code> ), but your LDAP administrator should verify that you're searching the right container.	<code>ou=users</code>
Object Class	Indicate the LDAP object class for user objects in the directory.	<code>inetOrgPerson</code>
First Name Attribute	Indicate the LDAP field that contains a user's first name.	<code>givenName</code>
Last Name Attribute	Indicate the LDAP field that contains a user's last name.	<code>sn</code>
Email Attribute	Indicate the LDAP field that contains a user's email address.	<code>mail</code>
<b>User subtree</b>	If enabled, subtrees of the Base DN are included when searching for users.	<b>Enabled</b>
<b>User ID attribute</b>	Indicate the LDAP field that should be identified as the User ID. This is the username to enter when logging in to contrast.	<code>uid</code>
<b>Authentication strategy</b>	Choose how you want Contrast to authenticate users when they provide their credentials. <b>Bind</b> means the application sends the user's credentials to the server for authentication. <b>Compare</b> means the server hashes the user's credentials and compares them to the value of the password attribute.	<b>Bind</b>

Option	Description	Default
<b>Password attribute</b>	The LDAP field that contains a user's password.  If you selected <b>Compare</b> for the authentication strategy, this attribute contains the hashed password for the user.	userPassword

- To automatically create new user accounts when someone makes an LDAP request to log in, check the box next to **Enable user provisioning**.  
Use the dropdowns to choose the **Default organization**, **Default organization role** and **Default application access group** for the new users.
- Contrast can automatically provision or de-provision users at login time based upon the user's LDAP groups. When this feature is enabled for LDAP-based authentication, users are added to a Contrast access group that maps to a corresponding LDAP group and removed from disallowed Contrast groups. Users can be added to multiple groups, as well as added to groups that give them access to multiple organizations.



### IMPORTANT

For this to work, the Contrast groups must already exist, and the groups from LDAP (for provisioning purposes) must have the same name as the Contrast groups.

- To add users to their groups when they log in to Contrast, check the box next to **Add users to their Contrast groups upon login**. To remove users from their groups when they log in to Contrast, check the box next to **Remove users from their Contrast groups upon login**.

## Use self-signed or privately-signed certificate with LDAP

If you [configure your LDAP integration \(page 1237\)](#) to connect to your server using SSL, you may need to import your server's certificate into a new truststore to be used by the Contrast JRE.

- To begin, acquire the server's certificate from your administrators in PKCS#12 format. If you're using a self-signed certificate, this is the actual LDAP server's certificate. If you have a private certificate (CA), you need the CA certificate for that server.
- Once you have the certificate for the server, import it into the truststore used by the JRE running Contrast. Run the following command as an administrator from a command shell in the directory where Contrast is installed.

```
$ jre/bin/keytool -import -file <path to certificate> -alias <hostname> \
-keystore <ts install>/jre/lib/security/cacerts
```

- You should now restart the Contrast server service, and verify that queries against LDAP now use SSL.

## Configure single sign-on (SSO) at a system level

Single sign-on (SSO) is an authentication service that allows access to multiple applications using one set of credentials. As a System Administrator, you can configure Contrast to use this service with a SAML 2.0 supported provider.

**NOTE**

- For more information, see the [SAML 2.0 specification](#).
- If you're not using SSO, protect your system by configuring [multi-factor authentication \(page 1232\)](#) as a required setting.
- If you configure SSO in Contrast and also want to use multi-factor authentication, configure it using your identity provider (IdP), instead of Contrast. With SSO configured, Contrast passes the responsibility of authenticating users to the IdP.

Authentication happens through an identity provider (IDP). You may use your own generic IDP or one of many popular third-party providers, such as [Okta](#), [OneLogin](#), [Ping Identity](#) or [ADFS](#).

Have your IDP metadata information ready, and then provide your metadata to connect to Contrast via an XML file or a Metadata URL.

For on-premises customers, the SuperAdmin configures SSO at the system level. Hosted customers can configure SSO at an organization level. Multi-tenant hosted instances can have multiple IDPs configured to a single instance of Contrast.

**NOTE**

If users are identified with a user ID rather than an email address, those accounts don't automatically transfer over to the SSO configuration and must be recreated.

## Before you begin

When using SSO, you must configure your `NameID` to pass the user's email.

Optionally, to set the user's first and last name, you must configure their IdP to pass additional attributes via the SAML assertion using:

- **First name:** `http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname`
- **Last name:** `http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname`

If those fields are not present or are blank, the default is to use the `NameID` field. And if user provisioning is enabled, the user's first and last name will auto-populate.

```
1<saml2:Attribute Name="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname"
2
3NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified"
4
5>
6 <saml2:AttributeValue xmlns:xs="http://www.w3.org/2001/XMLSchema"
7
8 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
9
10 xsi:type="xs:string"
11 >Dan</saml2:AttributeValue>
12 </saml2:Attribute>
```

## Steps

1. Contrast doesn't provide keys for SAML authentication. If you enable SSO without providing private keys, you're only able to perform IDP-initiated logins. You need to generate your own self-signed key using the Java Keytool:

```
keytool -genkeypair -alias some-alias -keypass changeit -keyalg RSA
-keystore samlKeystore.jks
```

2. Use the [encrypted properties editor \(page 1255\)](#) to modify *saml.properties*, and update the values to the keystore you created in the previous step.

```
authenticator.saml.keystore.path : /path/to/
samlKeystore.jks
authenticator.saml.keystore.default.key : some-alias
authenticator.saml.keystore.passwordMap : some-alias=changeit
authenticator.saml.keystore.password : changeit
```

3. Once you make the changes, restart Contrast so that it picks up the new keystore.
4. In the Contrast, under [system authentication \(page 1231\)](#), select **Authentication**, then **Change authentication method**.
5. Select **Single sign-on**.
6. Use the provided information to set up Contrast with your IDP. (You must also provide the **Entity ID** and **Metadata URL** in your IDP configuration.)



## Set Up Authentication

Step 2 Configure Your Identity Provider

Integrate your SAML Identity Provider (IdP) with Contrast to allow authentication of users using one set of credentials.

What you'll need: [Copy Metadata URL](#)

Entity ID *	Metadata URL	Attributes
http://ec2-52-68-234-25.ap-northeast-1.compute.amazonaws.com/Contrast/saml/metadata	http://ec2-52-68-234-25.ap-northeast-1.compute.amazonaws.com/Contrast/saml/metadata	None required
Version	NameID Format	Consumer Binding
2.0	Email Address	HTTP-POST
Assertion Consumer URL		
http://ec2-52-68-234-25.ap-northeast-1.compute.amazonaws.com/Contrast/saml/metadata		

What we need:

Identity Provider \*

☐ I have access to the metadata URL

IdP Metadata \*

```
<EntityDescriptor xmlns="urn:oasis:names:tc:SAML:2.0:metadata" entityID="http://ec2-52-68-234-25.ap-northeast-1.compute.amazonaws.com/Contrast/saml/metadata">
 <SPSSODescriptor AuthnRequestsSigned="false" WantAssertionsSigned="false"
 protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol
urn:oasis:names:tc:SAML:1.1:protocol http://schemas.xmlsoap.org/ws/2003/07/secext">
 <SingleLogoutService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
 Location="http://ec2-52-68-234-25.ap-northeast-1.compute.amazonaws.com/Contrast/saml/logout" />
 </SPSSODescriptor>
</EntityDescriptor>
```

Accepted Domain(s) (New Users) \*

[+ Add domain](#)

Default Organization:

Default Organization Role (New Users):

Default Application Access Group:

☒ Enable user provisioning

☐ Add users to their Contrast groups upon SSO login

☐ Remove users from their Contrast groups upon SSO login

- Provide a name for your **Identity provider**.
- Enter your IDP metadata. Select the box if you **have access to the metadata URL**, then enter the URL.
- If you want to automatically create new user accounts when someone make a SAML request to log in to Contrast, check the box next to **Enable user provisioning**.
  - Use the dropdowns to choose the **Default organization role** and **Default application access group** for the new users.
  - Add the **Accepted domains** that must be used to trigger user provisioning (for example, *yourdomain.com*).



### NOTE

You can also [automatically add users to groups \(page 1227\)](#).

- Select **Save**. If an error occurs, you can [check debug logs \(page 1210\)](#) for troubleshooting.
- [Restart Contrast \(page 1201\)](#) to apply the changes.

Once connected, you can return to the **SSO** tab to view and edit your settings. (You must retest and [restart Contrast \(page 1201\)](#) to apply the changes.) To return the organization back to the default configuration, select **Revert to Contrast-managed authentication** and confirm the change.

If SuperAdmin was disabled during installation, you're provided with two sets of metadata: one for the public node and one for the secret node. You need to set up the configuration for both in the Contrast interface.

## See also

- 🔗 [How to troubleshoot problematic SAML integrations](#)
- 🔗 [Configuring user and group provisioning with Okta](#)
- 🔗 [Configuring ADFS to automatically add users to groups](#)

## Enable HTTPS proxy authentication

You can use a trusted proxy for authentication, which authenticates the user and then sends the user's username to Contrast in an HTTP header. (This type of authentication is particularly useful for x509 clients.)

Users must be configured in Contrast before starting their authentication configuration, and use the same email address as their usernames for both configurations, in order to be granted access to Contrast after authentication.

To enable trusted HTTPS proxy authentication:

1. Update the *auth.properties* property file by changing the authentication mode in *~/path\_to\_contrast\_installation/data/conf/auth.properties* to *http\_header*.
2. By default, the HTTP header name is *Contrast-Authentication*. You can also configure this in the *auth.properties* file by updating the value of *auth.http.header.field.name*.
3. After restarting Contrast, each request must include the HTTP header *Contrast-Authentication: username* to log in.



### NOTE

Trusted HTTPS proxy authentication should only be used if all Contrast nodes are accessible exclusively through a trusted proxy. No nodes should be accessible directly; otherwise, an attacker could impersonate any authorized user.

## Set a password policy at a system level

Regulate passwords at a system level by creating a password policy.

1. Under [system settings \(page 1248\)](#), select **Security**.
2. Under **Default password policy**, select the box if you want to **Allow organization override**. This way organization administrators can [set password policy for their organizations \(page 1168\)](#).
3. Enter the following settings for your policy:
  - Set the password strength. This can be **Weak**, **Medium**, **Strong**, **Complex** or **Custom**. If you choose **Custom**, enter the number of minimum **Uppercase letters**, **Lowercase letters**, **Numbers** and **Symbols** required.
  - Enter the number of characters required in the **Minimum length** field.
  - Use the dropdown to choose the length of time allowed before **Password expiration**.
  - Enter the number of login attempts allowed before **Login logout**.
  - Enter the length of time allowed before **Inactive account expiration**.
  - Select the box to **Restrict password reuse**, and use the dropdown to choose the number of times each password may be reused.

- Select the box to **Restrict password reset**, and use the dropdown to choose the number of hours during which a user can reset their password after their reset request is sent.
- Use the dropdowns to select the amount of time that may pass before **Idle timeout** and **Session timeout**.

4. Select **Save**.

## Reset SuperAdmin password on Linux

To change the SuperAdmin password, edit the `contrast-server.vmoptions` file or use environment variables.

In most cases, editing the `contrast-server.vmoptions` file is the easiest method to use. If you are using containers for your Contrast application, use environment variables to reset the password.

### Steps

1. Open a command prompt and log in using the Contrast service account created during installation. For example:

```
sudo -su contrast_service
```

2. Shut down the Contrast server with this command:

```
$CONTRAST_INSTALLATION/bin/contrast-server stop
```

3. Are you going to use the `contrast-server.vmoptions` file to reset the password?

- If **yes**, go to [step 4](#).
- If **no**, go to [step 5](#).

4. To reset the password with the `contrast-server.vmoptions` file, use this procedure:

- a. Go to the `$CONTRAST_INSTALLATION/bin` directory. On most systems, this directory is `/opt/Contrast/bin`.
- b. Open the `contrast-server.vmoptions` file in a text editor.
- c. Add the following options to the file (replace `youremaildomain.com` with your email domain):

```
-Dreset.superadmin=true
-Dsuperadmin.username=contrast_superadmin@<your.email.domain.com>
-Dsuperadmin.password=<password>
```

5. To reset the password with environment variables instead of the `vmoptions` file, use this procedure:

- a. Go to the `$CONTRAST_INSTALLATION` directory. On most systems, this directory is `/opt/Contrast`.
- b. Enter the following command:

```
export INSTALL4J_ADD_VM_PARAMS="$INSTALL4J_ADD_VM_PARAMS
-Dreset.superadmin=true
-Dsuperadmin.username=contrast_superadmin@<your.email.domain.com>
-Dsuperadmin.password=<new password>"
```

6. Start the Contrast server with this command:

```
$CONTRAST_INSTALLATION/bin/contrast-server start
```

When the server starts, use the password you specified in step 4 or 5.

7. Shut down the Contrast server with this command:

```
$CONTRAST_INSTALLATION/bin/contrast-server stop
```

8. If you used the `contrast-server.vmoptions` file, remove the options you added in step 4.
9. Start the Contrast server as you normally would and exit the shell.

If you used environment variables to reset the password, this step clears the password from the `INSTALL4J_ADD_VM_PARAMS` environment variable.

## Reset SuperAdmin password on Windows

To reset the SuperAdmin password in the Contrast application.

1. Stop the Contrast Server service.
2. Launch a command prompt (`cmd.exe`) as an Administrator by right-clicking on **cmd.exe** and selecting **Run As Administrator**.
3. Go to the `Contrast\bin` directory. (On most systems, this is `C:\Program Files\Contrast\bin.`)
4. Use this command to edit the JVM options:

```
notepad contrast-server.vmoptions
```

5. Add the following options to the file. (Replace `youremaildomain.com` with your email domain.)

```
-Dreset.superadmin=true
-Dsuperadmin.username=contrast_superadmin@<your.email.domain.com>
-Dsuperadmin.password=<password>
```

6. Save the file and exit Notepad.
7. Use this command to start the Contrast service:

```
net start "Contrast Server"
```

8. Verify you are able to log in with the new password.
9. Enter the following command to stop the Contrast service:

```
net stop "Contrast Server"
```

10. Enter the following command to edit the JVM options:

```
notepad contrast-server.vmoptions
```

11. Remove the options added in step 5.
12. Save the file and exit Notepad.
13. Exit the command prompt.
14. Start the Contrast Server service as normal (from the Services control panel).

## Manage keys

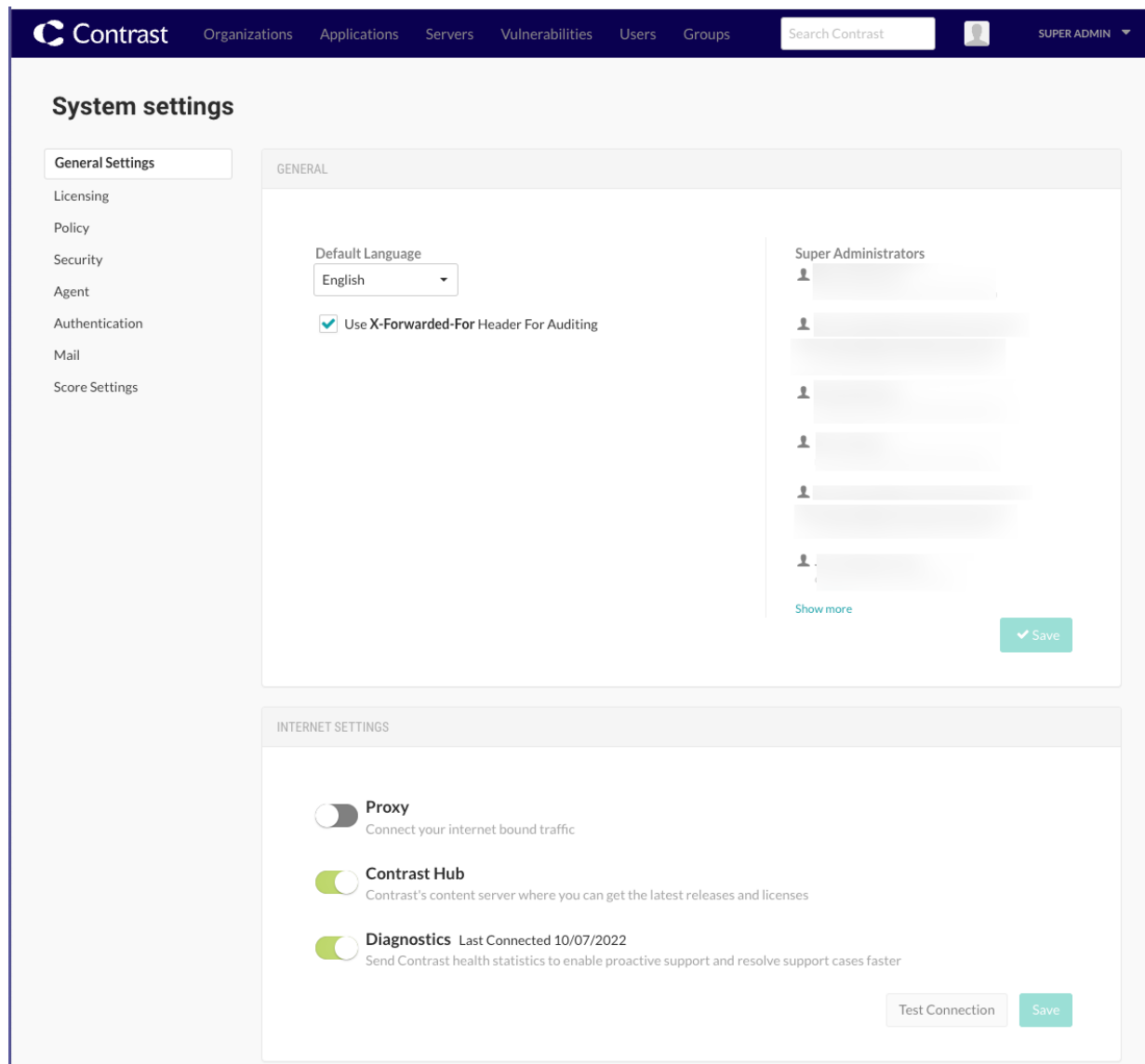
As a System Administrator, you can set policies for the keys across all your organizations to make sure they always meet your security standards.

1. In [system settings \(page 1248\)](#), select **Security**.
2. Under **API keys** choose the number of characters required as well as the minimum number of numerals, upper case characters and lower case characters required in the key. These standards apply when anyone [rotates the API key \(page 104\)](#).
3. Check the box at the top of the form if you want to **Mask invalid IPs on login**.
4. Select **Save**.

## Configure system settings

The System settings page lets you configure standard settings at a system level.

A [distributed deployment \(page 1197\)](#) of Contrast requires a different configuration.



## Steps

1. Log in as SuperAdmin, ServerAdmin or System Administrator.
2. Select **SuperAdmin** in the **user menu**.
3. Select **System settings** in the **user menu**.  
You can access:
  - [General settings \(page 1250\)](#)
  - [Licensing \(page 1251\)](#)
  - [Policy \(page 1254\)](#) (library compliance settings)
  - [Security \(passwords \(page 1246\), two-step authentication \(page 597\) and key management \(page 1248\)\)](#)
  - [Agent keys](#) (these are the same [Agent keys \(page 104\)](#) that you can find in Organization settings).
  - [Authentication \(page 1231\)](#)
  - [Mail \(page 1254\)](#)
  - [Score settings \(page 1177\)](#)

## Additional system settings

Additional system settings include:

- [Configure Tomcat \(page 1202\)](#)
- [Configure Java Runtime Environment \(JRE\) \(page 1203\)](#)
- [Configure HTTPS \(page 1203\)](#)
- [Add organizations \(page 1219\)](#)
- [Configure log levels \(page 1211\)](#)
- [Configure reporting storage \(page 1209\)](#)

## Configure general system settings

General settings is part of [system settings \(page 1248\)](#). Here, you can configure these settings:

- **Default language:** Select the language for your user interface.
- **X-Forwarded-For header:** Check the box if you want to use this header for auditing.
- **Proxy:** Connect your internet bound traffic.
- **Hub:** Integrate with Hub for library and CVE updates.
- **Diagnostics:** Send Contrast health statistics to enable proactive support and resolve support cases faster.
- **Agent diagnostics:** Send Contrast agent data to improve rules, performance and to prioritize product improvements.
- **Heroku settings:** Enter your password and SSO salt for the Heroku integration.

## Manage diagnostics at a system level

Contrast collects diagnostics that measure customer product usage to help provide faster, more proactive support and guide delivery of new functionality.

Contrast periodically sends snapshots of relevant data elements and aggregations to a diagnostics service on Contrast's hosted platform. Data that could be used to identify a customer or organization is obscured using a one-way hash, and is encrypted both in transit and at rest.

Diagnostic data provides aggregated insights, including counts and severity levels of detected vulnerabilities, Mean Time to Resolution (MTTRs), discovered and tested network routes, license utilization, and attack strings and IP addresses. This data is designed for high-level analysis and does not contain sensitive details such as personally identifiable information, source code, customer network identifiers, or specific vulnerability details beyond their general category and severity.

The data is then stored in a Contrast database, where it's made available to approved support and development users for analysis and reporting. Within the database, the data is attributed to customers to provide Customer Support insight into how to better assist Contrast customers.

Diagnostics improve customer support in three main ways:

- Customer Support analyzes diagnostics data for markers and reaches out proactively to customers to prevent problems.
- Data is used to quickly diagnose existing problems and reduce the cycle time for successfully resolving support cases.
- Deployment and usage insights help Contrast product development teams adjust and deliver new functionality that better address customer needs.

As a SuperAdmin, ServerAdmin or System Administrator, you can enable or disable diagnostics for your whole on-premises system:

1. In system settings, select **General settings** from the left navigation.
2. Under **Internet settings**, use the **Diagnostics** setting to disable or enable this feature. Diagnostics are enabled by default.  
Proxy settings apply to [Contrast Hub](#) and diagnostics settings.

**TIP**

You can also use the [REST API](#) to preview the data that will be transmitted to Contrast in these diagnostics.

## Allocate licenses at a system level

Contrast has these types of licenses:

- On-premises customers require a license to [install \(page 1190\)](#) and [upgrade \(page 1213\)](#) Contrast.
- Hosted and on-premises customers require specific licenses for Assess (application licenses), Protect (server licenses), and SCA (library licenses).

Use this procedure to allocate Assess and Protect licenses to organizations.

### Before you begin

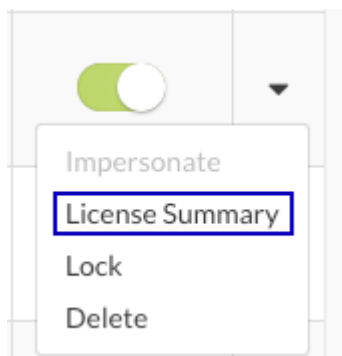
- For on-premises customers, a SuperAdmin or ServerAdmin role is required to allocate Assess and Protect licenses to a particular organization. Contrast Security handles this activity for hosted customers.
- Licenses applied to applications permanently count towards the number of maximum allowable applications. Deleting a licensed application has no effect on the number of licenses you are allowed to apply to applications.

### Steps to allocate licenses

1. Select **SuperAdmin** or **ServerAdmin** in the **user menu**.  
You see a list of organizations. The **Licenses** column shows the total number of Assess and Protect licenses available for each organization, followed by the number of unused licenses in parentheses.
2. You see a list of organizations. The **Licenses** column shows the total number of Assess and Protect licenses available for each organization, followed by the number of unused licenses in parentheses.
  - Select **Assess** or **Protect** in the Licenses column.



- Select the triangle ( ▾ ) at the end of a row and select **License summary**.



3. To add available Assess licenses to an organization:
  - a. Select **Add more licenses** above the Assess license bar.
  - b. In **Assess licenses**, enter the number of licenses you'd like to make available to this organization.
  - c. In **Expiration date**, enter an expiration date for the licenses you are adding.
  - d. Select **Allocate**.

**License summary**

A summary of licenses for the **AZTestOrg** organization.

**10 Assess licenses** [Add more licenses](#)

0/10

[Revoke unused licenses](#)

**Allocate Licenses** ×

[Go back](#)

Allocate licenses to the **AZTestOrg** Organization.

**Assess licenses** **Expiration Date**

450 02/01/2023 ×

999969157 of 999999999 Available

[Cancel](#) [Allocate](#)

4. To remove unused Assess licenses from an organization, select **Revoke unused licenses** below the Assess license bar.
5. To change the number of purchased Protect licenses allocated to an organization:
  - a. Select **Change license allocation** above the Protect license bar.
  - b. In **License allocation**, enter the number of licenses that you want to let the organization use. The default value is the number of purchased licenses. To revoke licenses for the organization, enter a value that's less than the number of purchased licenses. You can revoke more than the number of purchased licenses.
  - c. Select **Save changes**.



The image shows a 'License summary' modal for the 'AZTestOrg' organization. The modal is divided into two main sections: 'Assess licenses' and 'Protect licenses'. The 'Assess licenses' section shows a total of 10 licenses, with a progress bar at 0/10. It includes a link to 'Add more licenses' and a 'Revoke unused licenses' button. The 'Protect licenses' section shows a total of 10 licenses, with a progress bar at 0/10. It includes a 'Change license allocation' button. The modal also has a 'Go back' link and a 'Cancel' button.

**License summary**

A summary of licenses for the **AZTestOrg** organization.

**10 Assess licenses** [Add more licenses](#)

0/10

[Revoke unused licenses](#)

**10 Protect licenses** [Change license allocation](#)

0/10

**License summary**

[Go back](#)

**Allocate Protect licenses**

Change license allocation for the **AZTestOrg** organization.

**20 Protect licenses**

0/20

License allocation

20

[Cancel](#) [Save changes](#)

## Steps for license details and settings

1. To view the number of licenses available for each organization, Select **Organizations** in the header,  
The **Licenses** column shows the total number of Assess and Protect licenses available for each organization, followed by the number of unused licenses in parentheses.  
If Assess licenses are nearing expiration, you see a red warning icon. Hover over the icon to see the number of licenses expiring.
2. To view the number of available licences as well as the number of applications or servers that are unlicensed, from the user menu, select **System settings > Licensing**.
3. To automatically apply licenses to new applications or servers, under Licensing, select the toggle for Assess or Protect licenses. Then, select whether this setting applies to all applications or servers or only new ones.  
SuperAdmins, ServerAdmins and System Administrators also have the option to automatically apply licenses when you [add an organization \(page 1219\)](#).
4. In the top right of the Licensing window, you can select **Allow Organization Administrators to override** these settings (this setting defaults to enabled).

## Customize score settings at a system level

Contrasts designates an [application score \(page 1269\)](#), which can optionally depend on a [library score \(page 934\)](#). To customize score settings at a system level:

1. Under [system settings \(page 1248\)](#), select **Score settings**.
2. Select an option for **Overall score** to determine how applications are scored in Contrast:
  - **Default score** is the average of your application's [library score \(page 934\)](#) and its custom code score.
  - **Custom code-only score** ignores library score when calculating the overall application score. If you select this option, you can click to select specific languages, or apply it to all languages.
3. Select an option for **Library score** to determine how libraries are scored in Contrast:
  - **Default score** uses an algorithm that includes vulnerabilities as well as the age and versioning of a library.
  - **Vulnerability-only score** bases scoring solely on vulnerabilities present in the library.
4. Select the box next to **Allow organization override** so that an Organization Administrator can [determine score settings at an organization level. \(page 1177\)](#)
5. Select **Save**.



### NOTE

A RulesAdmin can configure policy settings in **Policy Management** so that any library in violation automatically receives a failing score (F). Once these settings are chosen, you'll see an alert message in Score Settings. Clicking the policy link in the alert navigates you to Library Policy, where administrators may view and revise these settings.

## Manage library compliance policy

You can manage library compliance policy at the system level. Library compliance policy allows you to restrict libraries that applications can safely use and set version requirements for specific libraries. Contrast can flag applications that use restricted libraries and flag or fail libraries that violate the compliance policy.

To manage library compliance policy:

1. Open the **user menu** (your name in the top right corner of Contrast) and select **System settings (page 1248)**.
2. Select **Policy**.
3. Set compliance requirements for your policy: libraries that are restricted from use, library version requirements, and whether Contrast should fail libraries that violate the policy.
4. Select **Allow organization override**, if you want Organization Administrators and RulesAdmins to [set compliance policy \(page 1147\)](#) at an organization level.

## Manage email notifications at a system level

System Administrators can enable or disable and configure Contrast to communicate with an appropriate SMTP system to receive these notifications.

Notifications allow Contrast users to receive alerts in specific situations, such as the discovery of a vulnerability or an attack on an application or when a password is reset.

Organization Administrators can [set default settings \(page 1175\)](#) for Contrast notifications at an organization level. Individual users can [adjust their own settings \(page 599\)](#).

To configure notifications at a system level:

1. Under [system settings \(page 1248\)](#), select **Mail** in the left navigation.
2. Configure these settings:
  - **Enable mail:** Use the toggle to enable or disable the feature.
  - **Mail protocol:** Values can be "SMTP" or "SMTPs".
  - **Mail host:** The fully qualified address of the SMTP server.
  - **Mail port:** The likely value is "25".
  - **Use SMTP auth:** Check the box to enable this setting.
  - **Mail user:** A user account for authentication purposes on the SMTP system.
  - **Mail password:** The password for the mail user associated with the SMTP system.
  - **Mail from:** Enter the email address you want system notifications to be sent from.
  - **Enable STARTTLS:** Check the box to enable this setting.
3. Select **Save**.

## Maintain Contrast on-premises

As a system administrator there are some ongoing tasks that are required for maintenance of the system. You may need to:

- [Back up MySQL databases \(page 1257\)](#)
- [Improve performance \(page 36\)](#)
- [Upgrade Contrast \(page 1213\)](#)
- [Upgrade the agents \(page 1214\)](#)
- [Update library data \(page 1215\)](#)
- [Update the Contrast license \(page 1217\)](#)
- [Update the IP address \(page 1215\)](#)
- [Manage SSL \(page 1259\)](#)
- [Use the encrypted properties editor \(page 1255\)](#)

## Use the encrypted properties editor

Contrast includes several configuration files in the `$CONTRAST_HOME/data/conf` directory. By default, Contrast encrypts the configuration files for security, but you can modify some of these files through workflows in Contrast.

For example, these are some of the encrypted properties files for on-premises installations:

Name	Contents
<i>ad.properties</i>	Settings to connect and configure Contrast to authenticate Active Directory groups.
<i>ldap.properties</i>	Settings to connect and configure Contrast to authenticate LDAP groups.
<i>database.properties</i>	Host and connection settings for communication between Contrast and MySQL.
<i>saml.properties</i>	SAML keystore security settings.

Contrast also includes an editing tool to decrypt these files and assist with configuration. This is helpful when you are [running Contrast \(page 1199\)](#) and need to get values from encrypted properties files outside of the application or automatically update a property in the files, such as automatic password rotation.

To edit encrypted properties files:

1. Find the decryption tool in the `$CONTRAST_HOME/bin` directory.
  - **Linux:** the file is a shell script called `edit-properties`.
  - **Windows:** the file is a Windows command file called `edit-properties.exe`.

2. Run the tool from a command prompt. This opens an application that allows you to update the value of an encrypted property:

```
$CONTRAST_HOME/bin/edit-properties -e $CONTRAST_HOME/data/esapi -f
$CONTRAST_HOME/data/conf/ad.properties
```

3. You must provide input details to view or edit encrypted properties files. The basic inputs you need are:

- The path to *ESAPI.properties*.
- The target properties file to edit.

To find this information for the encrypted properties editor, execute `edit-properties` with no arguments:

```
contrast@EOP-TeamServer:~/contrast/bin$./edit-properties

usage: property-editor
-c,--comment <text> The comment for the top of the file
-e,--esapi <path> The path to the ESAPI.properties file
-f,--targetFile <file> The properties file to edit
-o,--print-value Print out the value of the property and exit
-p,--property <name> The name of the property to set
-v,--value <val> The value of the property
```

4. This example shows you how to edit an encrypted file. Provide the path to *ESAPI.properties* and the target properties file to edit. You will see the existing values encrypted in the file that you can edit. The usage options above allow you to view or edit a single property.

```
contrast@TeamServer:~/contrast/bin$./edit-properties -e ../data/esapi/
-f ../data/conf/ad.properties

ad.userDn : cn=Directory Manager
ad.identity.attribute.name : mail
ad.password : NotaRealPassword
ad.nested.groups.enabled : false
ad.group.users :
cn=ContrastUsers,cn=Users,dc=contrastsecurity,dc=com
ad.group.admin :
cn=ContrastAdmins,cn=Users,dc=contrastsecurity,dc=com
ad.url : ldap://localhost:389
ad.base :
dc=contrastsecurity,dc=com
```

5. You can also retrieve or update unencrypted values for a property. To retrieve values, pass another parameter to the properties editor. In this example, the user is looking for details about database properties:

```
$CONTRAST_HOME/bin/edit-properties \
-e $CONTRAST_HOME/data/esapi \
-f $CONTRAST_HOME/data/conf/database.properties \
-p jdbc.username \
-o
```

To update unencrypted values, pass a different set of arguments to the properties editor:

```
$CONTRAST_HOME/bin/edit-properties \
-e $CONTRAST_HOME/data/esapi \
-f $CONTRAST_HOME/data/conf/database.properties \
-p jdbc.username \
-v joe.user \
-c "Updating JDBC Password"
```

**NOTE**

Add comments to indicate edits to encrypted properties files. This is useful for auditors or others who need to track configuration changes.

## MySQL backups

Use these procedures to maintain the MySQL databases that the Contrast installer creates.

These procedures do not apply to a MySQL database that you create for distributed environments.

- [Create an automated MySQL backup \(page 1257\)](#)
- [Backup MySQL manually \(page 1257\)](#)
- [Restore database backups \(page 1258\)](#)
- [Disable automated backups \(page 1258\)](#)

## Backup MySQL manually

You can also use the `backup-db` script included with Contrast to do this.

### Before you begin

- Be sure you have permission to run the `backup-db` script. Typically, you must be the installation owner for a Contrast, root or Windows Administrator account to do this.
- Be sure that Contrast is running and MySQL is available.

### Steps

1. If you have not done so, configure the database backup location. Set a location for `database.bk.dir` by editing the `$CONTRAST_HOME/data/conf/database.properties` file with the [encrypted editor \(page 1255\)](#).
2. Run the backup command for your environment:
  - **Windows:** `$CONTRAST_HOME\bin\backup-db.cmd`
  - **Linux:** `$CONTRAST_HOME/bin/backup-db.sh`

## Create an automated MySQL backup

You can create a backup of the Contrast MySQL database on a regularly scheduled basis. During [installation \(page 1190\)](#), you can select this option and define a time and location for storing database backups.

If you skip this step during installation, you can still configure Contrast later to schedule database backups.

### Steps

1. Find Contrast database settings in `$CONTRAST_HOME/data/conf/database.properties`.
2. [Use the encrypted properties editor \(page 1255\)](#) to identify database settings. The example below shows a Contrast database with backups enabled, scheduled and in a specific location. You can edit these settings, if any options need to change.

```
contrast@TeamServer:~/contrast/bin$./edit-properties -e ../data/esapi/
-f ../data/conf/database.properties

database.bk.time : 4:0:0
database.bk.enabled : true
```

```
database.bk.dir : /mnt/backups/mysql/
contrast
```

3. If you want to upgrade Contrast, you should capture any data created or changed since the last scheduled backup.

## Disable automated backups

You can stop automated backups of the Contrast MySQL database. Scheduled backups run through `schtasks` on Windows and `crontab` on Linux.

To disable automated backups:

For **Windows**, use Task Scheduler to disable or delete `ContrastBackup`.

For **Linux**:

1. Switch to the user under which you installed Contrast and run `crontab -l`.
2. This lists the scheduled job. You will see:

```
0 2 * * * /usr/local/contrast/bin/backup-db.sh
```

3. Run `crontab -e` to delete a single backup. Run `crontab -r` to delete all backups.



### CAUTION

The `-e` option allows edits with Vim to delete selected backups. The `-r` option deletes everything: be careful when you use it.

## Restore database backups

Use this procedure to restore the MySQL database that the Contrast installer creates.

This procedure does not apply to MySQL databases that you create for distributed environments.

### Before you begin

Database restoration should be performed by a MySQL Database Administrator.

### Steps

1. Use the [encrypted properties editor \(page 1255\)](#) to identify the MySQL database settings.
2. Shut down Contrast.
3. Start up MySQL individually using the MySQL service packaged with Contrast. Replace `<YourPath>` with the path to your Contrast home.

- **Windows:**

```
"<YourPath>\mysql\bin\mysqld.exe" --defaults-
file="<YourPath>\data\conf\my.cnf"
```

- **Linux:**

```
sudo -u contrast_service <YourPath>/mysql/bin/mysqld --
defaults-file=<YourPath>/data/conf/my.cnf --basedir=<YourPath>/mysql
--datadir=<YourPath>/data/db --plugin-dir=<YourPath>/mysql/lib/plugin
--lc-messages-dir=<YourPath>/mysql/share --tmpdir=/tmp --lc-
messages=en_US --log-error=<YourPath>/logs/mysql_error.log --pid-
file=<YourPath>/data/proc/MysqldResource.pid --port=13306
```

4. Connect to MySQL. Replace `<jdbc.host>`, `<jdbc.port>`, `<jdbc.user>` and `<jdbc.schema>` with your host, port, user and schema.

- **Windows:**

```
mysql -h <jdbc.host> -P <jdbc.port> -u <jdbc.user> -p <jdbc.schema>
```

- **Linux:**

```
./mysql -h <jdbc.host> -P <jdbc.port> -u <jdbc.user> -p <jdbc.schema>
```

5. Drop the Contrast database with `drop database <jdbc.schema>;`.
6. Create the Contrast database with `create database <jdbc.schema>;`.
7. Grant permissions to the Contrast user with `GRANT ALL PRIVILEGES ON *.* to 'contrast'@'%' ;`.
8. Exit from MySQL.
9. Restore the MySQL backup. Replace `<backup_location>` with your backup location and `<backup_filename>` with your backup filename.

- **Windows:**

```
mysql -h <jdbc.host> -P <jdbc.port> -u <jdbc.user> -p <jdbc.schema> <
<backup_location>/<backup_filename>
```

- **Linux:**

```
./mysql -h <jdbc.host> -P <jdbc.port> -u <jdbc.user> -p <jdbc.schema>
< <backup_location>/<backup_filename>
```

10. Shut down MySQL:

- **Windows:**

Use the Windows Service Manager application to shut down the MySQL service.

- **Linux:**

```
$CONTRAST_HOME/mysql/bin/mysqladmin.exe shutdown -h localhost -P 13306
-u contrast -p
```

You are prompted for the password that is set in the encrypted properties editor.

11. Restart the fully-restored Contrast and MySQL together.

## Manage SSL

On-premises customers may need to use a Secure Sockets Layer (SSL) in the following situations:

- Setting up [HTTPS proxy authentication \(page 1246\)](#)
- Integrating with [LDAP \(page 1237\)](#) or [Active Directory \(page 1233\)](#)
- Securing communication between agents and the Contrast application

# Reference

These topics may be useful as an occasional reference on how to use Contrast:

- [Glossary \(page 1260\)](#)
- [Roles and permissions \(page 1264\)](#)
- [Agent supported technologies \(Java \(page 120\), .NET Framework \(page 212\), .Net Framework \(Legacy\), .NET Core \(page 271\), Node.js \(page 332\), Python \(page 410\), Ruby \(page 469\), Go \(page 530\)\)](#)
- [Application scoring guide \(page 1269\)](#)
- [Library scoring guide \(page 934\)](#)
- [Log levels \(page 1271\)](#)
- [Supported browsers \(page 1275\)](#)
- [Regular expressions for application exclusions \(page 1123\)](#)

## Glossary of terms

These terms are defined specifically as they apply to users of Contrast. You can hover over these terms in other topics to pull up the definition in context.

account takeover (ATO)	An account takeover is the result of an attack that steals login credentials or otherwise breaks authentication in web applications.
agent	An agent is language-specific code that is installed in a web application to gather and analyze security data, and report findings to Contrast when necessary.
application	An application is a logical grouping of customer code analyzed by a Contrast agent.
attack	An <a href="#">attack (page 584)</a> is made up of one or more attack events that occur within a discrete time frame.
attack event	An attack event is a violation of Protect rules or other suspicious application activity in instrumented applications. The event corresponds to a single attack vector, such as an HTTP request or SQL query. Multiple attack events make up an attack, usually in the same area of code and timeframe.
brute-force attack	A brute-force attack is the systematic submission of many passwords or passphrases with the intent of eventually guessing correctly.
chief information security officer (CISO)	The chief information security officer directs an organization's information security program to assure and demonstrate that sensitive assets are well-protected and staff can manage and prevent vulnerabilities.
command injection	Command injection attacks target the host operating system through a vulnerable application. They happen when a user passes unsafe data to a system shell through a form, cookie or HTTP header or some other part of the application.
common language runtime (CLR)	Common Language Runtime manages the execution of .NET Framework programs.



Common Vulnerabilities and Exposures (CVE)	<a href="#">Common Vulnerabilities and Exposures</a> is a list of publicly known cybersecurity vulnerabilities, used internationally to identify and track types of vulnerabilities.
container image	A container image is a static file with executable code that can create a container on a computing system.
continuous integration/continuous delivery (CI/CD)	Agile practice that encourages continuous iterations and automation in building, testing and deployment.
Contrast command line interface (Contrast CLI)	The Contrast CLI is a text-based user interface. It can be run in the development environment to get early software composition analysis (SCA) visibility of your open-source libraries before you build and deploy. Results from the CLI can be viewed in the text-based response and they are also represented as a <a href="#">dependency tree (page 934)</a> in Contrast.
Contrast Hub	The <a href="#">Contrast Hub</a> is where on-premises customers can download the Contrast installer and license files and check for agent updates.
Contrast service	The Contrast service is a program written in Go that connects the Contrast web interface with the Node.js, Ruby and Python agents.
credential stuffing	Credential stuffing is a brute force attack that automatically injects pairs of breached usernames and passwords to access user accounts.
cross-site scripting (XSS)	Cross-site scripting is an attack that occurs when malicious scripts are injected into a web application through user inputs that generate output without validating or encoding it.
dependency confusion	Dependency confusion, also known as a "substitution attack," is when an attacker registers the same name for an organization's internal library on a public package index in order to send vulnerable or malicious code into the organization's private code repositories.
distroless image	Distroless images contain only your application and its runtime dependencies. They do not contain package managers, shells, or any other programs you would expect to find in a standard Linux distribution.
dynamic application security testing (DAST)	A security testing technology that is designed to detect conditions that point to a security vulnerability in an application in its running state.
environment	In Contrast, applications are organized into one of three environments: development, test (QA) and production.
environment variable	Environment variables are values you can pass to software at runtime, usually key/value pairs that you define outside of an application. In Contrast, these are used to <a href="#">configure the agents (page 102)</a> that instrument your applications and make sure they work within your preferred frameworks as expected and report metadata you want to see in Contrast.
Exploit Prediction Scoring System (EPSS)	<p>Calculation based on the likelihood of a vulnerability being exploited.</p> <p>EPSS provides a probability range between 0 to 1 (0 and 100%). A higher score indicates a vulnerability likely will be exploited within 30 days.</p>

	<p>The EPSS percentile is a percentage score assigned to a specific vulnerability that indicates how likely it is to be exploited compared to other vulnerabilities. For instance, a vulnerability with an EPSS percentile of 90% means it has a higher probability score than 90% of all other CVEs in the group.</p>
false positive	<p>A vulnerability that is falsely reported.</p>
flow map	<p>A flow map is a visualization of an instrumented application in Contrast that shows all back-end systems it uses and any other applications connected to it. This helps you assess risk by analyzing what else touches vulnerable applications.</p>
instrument	<p>Monitor applications with software agents that observe and report data at runtime. Contrast agents send security vulnerability data about your applications based on exercised routes.</p>
interactive application security testing (IAST)	<p>Security technology that analyzes data flows within a running application to detect and report possible security vulnerabilities.</p>
IP allowlist	<p>An IP allowlist is a rule that allows any HTTP request from IP addresses on that list.</p>
IP denylist	<p>An IP denylist is a rule that blocks any HTTP request from IP addresses on that list.</p>
library	<p>A library is any packaged code included in an application. <a href="#">Libraries (page 922)</a> can be public or private.</p>
lightweight directory access protocol (LDAP)	<p>LDAP is a lightweight client-server protocol for accessing and maintaining directory services. In Contrast, on-premises can <a href="#">use LDAP (page 1237)</a> to manage users and logins.</p>
manifest	<p>Files that are stored with a project to declare which dependencies are required by a project.</p>
National Institute of Standards and Technology (NIST)	<p>NIST is a <a href="#">government agency</a> that promotes U.S. innovation and industrial competitiveness by advancing measurement science, standards, and technology, in multiple fields including cybersecurity.</p>
Not a problem	<p><i>For library status.</i> This library has vulnerabilities that are acknowledged and the risks are acceptable.</p>
Open Web Application Security Project (OWASP)	<p><a href="#">Open Web Application Security Project®</a> is a nonprofit foundation that works to improve the security of software through an open platform that supports shared security projects and member education. The <a href="#">OWASP Top Ten</a> lists the most critical security defects in Web applications.</p>
path traversal	<p>Path traversal is an attack that attempts to access critical system files and directories stored outside the web root folder. It uses variables that reference files with "dot-dot-slash" (../) sequences or absolute file paths.</p>
policy	<p>A policy is the set of rules for a given application or library that triggers security violations, status changes, or notifications when certain conditions are true. Policies assure more consistent security standards across applications or teams.</p>

profiler chaining	Profiler chaining is a way to run the .NET Framework or .NET Core agent alongside other .NET profiler agents, such as performance or APM tools.
Remediated	<i>For library status.</i> The vulnerable library has been remediated.
Reported	<i>For library status.</i> Status when a library with vulnerabilities is detected by Contrast.
Route	A route, as reported in Contrast Assess, is a function signature of the method that handles requests to one or more URL patterns. The exact format can vary, depending on the language of the Contrast agent you are using. A route can have multiple URLs associated with it.
rule	A rule is a security control used to identify a vulnerability or attack event. If the rule matches, the agent sends a vulnerability or attack event to Contrast for the affected application.
runtime application self-protection (RASP)	Contrast uses RASP methods to monitor attacks and actively defend applications in production.
score	Contrast provides <a href="#">library (page 934)</a> and <a href="#">application (page 1269)</a> scores that reflect the current security situation for your applications or libraries.
security assertion markup language (SAML)	SAML is an XML-based standard used to create and exchange security information between online partners, such as single sign-on authentication.
security incident event management (SIEM)	SIEM systems collect and analyze security events and related data from other systems to support threat detection, compliance, and security incident management. Contrast integrates with several leading SIEM systems.
sensitive data masking	This feature redacts sensitive data in Contrast logs and other data transmissions from the Contrast agent, without affecting how that data is processed by the application.
single sign-on (SSO)	Single sign-on is an authentication or user identification service that gives users access to multiple systems with only one set of credentials.
sink	In <a href="#">data flow analysis (page 1032)</a> , the sink is where data ends. A sink is any external format or location to which data is written.
Software Composition Analysis (SCA)	Identify vulnerable libraries, fail a build based on CVE severity, and view a dependency tree to understand the dependencies between libraries and where vulnerabilities have been introduced.
software development life cycle (SDLC)	The series of steps by which ideas become software that is used by people.
source	In <a href="#">data flow analysis (page 1032)</a> , the source is where data starts. A source is any input data or request that enters a system.
SQL injection (SQLi)	A SQL Injection attack inserts or "injects" a SQL query within user input data from the client to get it into the application. The intent is to read or modify database data or send commands to the database ( <a href="#">for example</a> ).

stack trace	A stack trace lists the sequence of events that led to a failure. For Contrast, the stack trace shows the events that led to a security vulnerability.
static application security testing (SAST)	A method of finding potential vulnerabilities in applications without installing or running the application.
testing coverage	This is a testing technique that monitors the number of tests that have been executed.
unused functions	Also called shadow functions. This is a function on a cloud environment that has not been invoked for over 90 days.
web application firewall (WAF)	A WAF inspects and filters web traffic to defend applications from common attacks.
webhook	Integration method that sends real-time data from one application to another via HTTP every time a specified event occurs.

## Open source software attributions

Contrast software includes source code from these open source software packages:

Source code	Version	License
<a href="#">html2canvas</a>	V1.4.1	<a href="#">MIT license</a>
<a href="#">jsPDF</a>	V2.5.1	<a href="#">MIT license</a>
<a href="#">bubkoo/html-to-image</a>	V1.11.1	<a href="#">MIT license</a>

## Roles and permissions

Permissions and capabilities are granted to users depending on the role they are assigned. Roles exist at the [application \(page 1264\)](#), [organization \(page 1267\)](#) and (for on-premises customers) [system \(page 1269\)](#) levels. Most of these roles are defined when a user is assigned to an [access group \(page 1164\)](#).

You can also:

- [View your own permissions \(page 599\)](#)
- [Manage organization permissions \(page 1163\)](#)
- [Manage system permissions \(page 1218\)](#)
- [Grant Protect permissions \(page 1226\)](#)



### NOTE

API-Only users can access Contrast's REST API, but can't log in to the user interface. Contrast doesn't recommend the creation of administrator API accounts.

## Application roles

Application roles give users permissions and capabilities within a particular application. Application roles are assigned to [access groups \(page 1164\)](#).

Use these application roles to grant permissions and capabilities within an application:

## View

The application View role (Application Viewer) has read-only access to the Contrast interface to see scores, libraries, vulnerabilities and comments, but cannot perform edits to traces to the application.

### View Permission Details (Application)

#### Application

- ☐ Archive
- ☐ Delete
- ☐ Edit
- ☐ License
- ☐ Merge
- ☐ Reset
- ☐ Restore
- ☒ Tag
- ☒ View

#### Vulnerabilities

- ☐ Delete
- ☐ Discussion
- ☐ Edit
- ☒ Export
- ☒ HTTP Replay
- ☐ Merge
- ☐ Send to Bugtracker
- ☒ Tag
- ☒ View

#### Libraries

- ☒ Manifest
- ☒ Tag
- ☒ View

#### Policy

- ☐ Edit
- ☒ View

#### Exclusions

- ☐ Create
- ☒ View

#### Report

- ☐ Generate

#### Architecture

- ☐ View

Done

## Edit

The application Edit role (Application Editor) can remediate findings, add tags, manage vulnerabilities, edit attributes, merge applications, add or delete applications, and create servers. The majority of Contrast users have this role.

### Edit Permission Details (Application)

#### Application

- ☒ Archive
- ☐ Delete
- ☐ Edit
- ☐ License
- ☒ Merge
- ☐ Reset
- ☒ Restore
- ☒ Tag
- ☒ View

#### Vulnerabilities

- ☒ Delete
- ☒ Discussion
- ☒ Edit
- ☒ Export
- ☒ HTTP Replay
- ☒ Merge
- ☒ Send to Bugtracker
- ☒ Tag
- ☒ View

#### Libraries

- ☒ Manifest
- ☒ Tag
- ☒ View

#### Policy

- ☐ Edit
- ☒ View

#### Exclusions

- ☐ Create
- ☒ View

#### Report

- ☒ Generate

#### Architecture

- ☐ View

Done

## Rules Admin

The application Rules Admin role (Application Rules Admin) can edit rules and policies in the application, enable Protect, and manage notifications and scoring for the application.

### Rules Admin Permission Details (Application)

#### Application

- ☒ Archive
- ☐ Delete
- ☐ Edit
- ☐ License
- ☒ Merge
- ☐ Reset
- ☒ Restore
- ☒ Tag
- ☒ View

#### Vulnerabilities

- ☒ Delete
- ☒ Discussion
- ☒ Edit
- ☒ Export
- ☒ HTTP Replay
- ☒ Merge
- ☒ Send to Bugtracker
- ☒ Tag
- ☒ View

#### Libraries

- ☒ Manifest
- ☒ Tag
- ☒ View

#### Policy

- ☒ Edit
- ☒ View

#### Exclusions

- ☒ Create
- ☒ View

#### Report

- ☒ Generate

#### Architecture

- ☒ View

Done

## Admin

The application Admin role (Application Administrator) has no restrictions and can manage other users' access to the application.

### Admin Permission Details (Application)

#### Application

- ☒ Archive
- ☒ Delete
- ☒ Edit
- ☒ License
- ☒ Merge
- ☒ Reset
- ☒ Restore
- ☒ Tag
- ☒ View

#### Vulnerabilities

- ☒ Delete
- ☒ Discussion
- ☒ Edit
- ☒ Export
- ☒ HTTP Replay
- ☒ Merge
- ☒ Send to Bugtracker
- ☒ Tag
- ☒ View

#### Libraries

- ☒ Manifest
- ☒ Tag
- ☒ View

#### Policy

- ☒ Edit
- ☒ View

#### Exclusions

- ☒ Create
- ☒ View

#### Report

- ☒ Generate

#### Architecture

- ☒ View

Done

## Auditor

The application Auditor role can only access the Contrast API, not the Contrast application. Users with this permission can only access the V3 audit API. [RBAC permissions \(page 1277\)](#) access the V4 API.

### Auditor Permission Details (Application)

#### Application

#### Vulnerabilities

#### Libraries

#### Policy

#### Report

#### Exclusions

#### Architecture

Done

The **No Access** role blocks user access to the application.

You can add application roles when you [create or edit an organization access group](#) (page 1164).

## See also

[View permissions](#) (page 599)

## Organization roles

Users may have different roles across different organizations.

Every user has a default role for the default organization.

These are the organization roles:

**View**

The View role for organizations (Organization Viewer) has read-only access to the Contrast interface to see scores, libraries, vulnerabilities, and comments but cannot edit traces to the application.

**View Permission Details** (Organization) ×

**General**

- ☒ Org Info
- ☐ Report Settings
- ☒ Score Settings

**Servers**

- ☐ Delete
- ☐ Edit
- ☐ License
- ☒ Tag
- ☒ View

**Security**

- ☐ Audit
- ☒ Get API Keys
- ☐ IP Range
- ☐ Rotate API Keys
- ☐ Password Policy
- ☐ Session Timeouts

**Integrations**

- ☐ Edit
- ☐ View

**Libraries**

- ☒ Manifest
- ☒ Tag
- ☒ View

**Policy**

- ☐ App Exclusions
- ☐ Assess Rules
- ☐ Library Policy
- ☐ Protection Policy
- ☐ Remediation Policy

Done

**Edit**

The Edit role for organizations (Organization Editor) can remediate findings, add tags, manage vulnerabilities, edit attributes, merge applications, add or delete applications, and create servers. The majority of Contrast users have this role.

**Edit Permission Details** (Organization) ×

**General**

- ☒ Org Info
- ☐ Report Settings
- ☒ Score Settings

**Servers**

- ☐ Delete
- ☐ Edit
- ☐ License
- ☒ Tag
- ☒ View

**Security**

- ☐ Audit
- ☒ Get API Keys
- ☐ IP Range
- ☐ Rotate API Keys
- ☐ Password Policy
- ☐ Session Timeouts

**Integrations**

- ☐ Edit
- ☐ View

**Libraries**

- ☒ Manifest
- ☒ Tag
- ☒ View

**Policy**

- ☐ App Exclusions
- ☐ Assess Rules
- ☐ Library Policy
- ☐ Protection Policy
- ☐ Remediation Policy

Done

**Rules Admin**

The Rules Admin role for organizations (Organization Rules Admin) can edit rules and policies in the application, enable Protect, and manage notifications and scoring for the organization.

**Rules Admin Permission Details** (Organization) ×**General**

- ☒ Org Info
- ☐ Report Settings
- ☒ Score Settings

**Servers**

- ☐ Delete
- ☐ Edit
- ☐ License
- ☒ Tag
- ☒ View

**Security**

- ☐ Audit
- ☒ Get API Keys
- ☐ IP Range
- ☐ Rotate API Keys
- ☐ Password Policy
- ☐ Session Timeouts

**Integrations**

- ☐ Edit
- ☐ View

**Libraries**

- ☒ Manifest
- ☒ Tag
- ☒ View

**Policy**

- ☒ App Exclusions
- ☒ Assess Rules
- ☒ Library Policy
- ☒ Protection Policy
- ☒ Remediation Policy

Done

**Admin**

The Admin role for an organization (Organization Administrator) is responsible for the configuration and management of the organization.

**Admin Permission Details** (Organization) ×**General**

- ☒ Org Info
- ☒ Report Settings
- ☒ Score Settings

**Servers**

- ☒ Delete
- ☒ Edit
- ☒ License
- ☒ Tag
- ☒ View

**Security**

- ☒ Audit
- ☒ Get API Keys
- ☒ IP Range
- ☒ Rotate API Keys
- ☒ Password Policy
- ☒ Session Timeouts

**Integrations**

- ☒ Edit
- ☒ View

**Libraries**

- ☒ Manifest
- ☒ Tag
- ☒ View

**Policy**

- ☒ App Exclusions
- ☒ Assess Rules
- ☒ Library Policy
- ☒ Protection Policy
- ☒ Remediation Policy

Done

**Auditor**

The organization Auditor role can only access the Contrast API, not the Contrast application. Users with this permission can only access the V3 audit API. [RBAC permissions \(page 1277\)](#) access the V4 API.

**Auditor Permission Details** (Organization) ×**General****Servers****Security****Integrations****Libraries****Policy**

Done

You assign organization roles by [adding users to an organization access group \(page 1164\)](#).



## See also

[View permissions \(page 599\)](#)

## System roles



### NOTE

These roles are only available to on-premises customers.

When deciding how to [manage system administration \(page 1218\)](#), you can use these roles to assign permissions and capabilities:

- A **SuperAdmin** is responsible for the installation and setup of on-premises instances. They may also be responsible for the system administration of Contrast, however, this may be assigned to one or more System Administrators. **SuperAdmins** can configure organizations, applications, servers, vulnerabilities, users and groups.
- A **ServerAdmin** is identical to a SuperAdmin except without access to users or groups. They have access to the **ServerAdmin** option in the user menu, which allows them to configure organizations, applications, servers and vulnerabilities.
- A **System Administrator** is responsible for maintaining organizations and groups. They have access to the **SuperAdmin** option in the user menu, which allows them to configure organizations, applications, servers, vulnerabilities, users and groups.
- A **System Observer** has read-only access to organizations, users, applications, groups and traces. They have read-only access to the **Observer** option in the user menu, which allows them to view organizations, applications, servers, vulnerabilities and users.
- The **No Access** role for a particular organization blocks users from that organization.

## Application scoring guide

The application score helps you gauge the general performance of each application.

Scores are based on how much of the application has been exercised, as well as the amount and severity of vulnerabilities found for that application.

Numeric scores map to letter grades that are shown in Contrast:

- **A:** 90-100
- **B:** 80-89
- **C:** 70-79
- **D:** 60-69
- **F:** 35-59

To calculate the application score, find the average of the application's [library score \(page 934\)](#) and the custom code score.

To calculate the custom code score, start with 100 points and subtract penalty points for the number of vulnerabilities found in your application times a penalty weight for their severity, shown here:

- **Critical:** Multiply the number of vulnerabilities by 20
- **High:** Multiply the number of vulnerabilities by 10
- **Medium:** Multiply the number of vulnerabilities by 5

- **Low:** Multiply the number of vulnerabilities by 1

Vulnerabilities are weighted differently depending on how likely they are to be exploited and how serious the effects would be.

For example, a SQL injection is considered **Critical** because automated tools exist to exploit them without expertise. An attacker who doesn't know anything about your application or schema can exfiltrate your entire database contents.

On the other hand, using a hashing algorithm like SHA-1 is considered **Low** because it has been known to exhibit serious weaknesses. Also it requires the resources of a very skilled attacker with extensive backing.



### TIP

For example, to calculate your application score:

First determine your custom code score. If your application had 0 **Critical**, 1 **High**, 2 **Medium** and 1 **Low** vulnerability, your custom code score would be:

$$100 - (20 \times 0) - (10 \times 1) - (5 \times 2) - (1 \times 1) = 79$$

If you are running Contrast on an application with a library score of 85 and a custom code score of 79, your application score would be 82 which would be a B.

$$85 + 79 = 164$$

$$164 / 2 = 82$$

To improve your score:

- [Enable Protect rules \(page 1130\)](#) and CVE shields to remove protected vulnerabilities from the score calculation.
- Remediate **Critical** and **High** vulnerabilities in your custom code.
- Address the vulnerable libraries.
- Update **High risk** libraries.

## Library scoring guide

Contrast provides letter grades for the security of your application's libraries so that you can use them as a reference point during analysis. The grades map to scores as follows:

- A: 90 - 100
- B: 80 - 89
- C: 70 - 79
- D: 60 - 69
- F: 35 - 59

Scores are based on three penalty factors:

- **Time:** The age of the library is calculated based on the number of full years between the release of the latest version and the version used in the application, multiplied by 2.5.
- **Status:** The status is calculated based on the number of versions that have been released since the current library in your application, multiplied by 10.

- **Security:** The CVE penalty of the library is the highest severity of all known CVEs for this library, multiplied by 10.

**NOTE**

Organization administrators can [adjust the scoring method \(page 1177\)](#) to include only security criteria.

**TIP**

For example:

If you're using a library from January 2010 and the latest version came out in September 2013, the number of full years passed is two. So your time penalty would be:

$$2 \times 2.5 = 5$$

If you're using Version 1.1.1, but Versions 1.1.2 and 1.1.3 have been released, your penalty would be:

$$2 \times 10 = 20$$

If you have a library with the scores 2.4 and 2.2, the penalty would be:

$$2.4 \times 10 = 24$$

The final score of the library is calculated by subtracting each of the three penalty values from 100.

$$100 - 5 - 20 - 24 = 51$$

A score of 51 maps to a letter grade of F.

## Log levels

The log level setting controls which events are processed by server logging, and can help you more effectively capture events. They can be [configured at a system level \(page 1211\)](#) or for a [particular server \(page 915\)](#) by any user with Editor permissions.

Log levels follow the Log4j standard and honor their level designations as much as possible.

INFO is the default value. ERROR level is sufficient in most cases, unless a problem occurs and you need to collect more detailed metrics.

Log level	Description
ERROR	Gives information about a serious error that needs to be addressed and may result in an unstable state.
WARN	Gives a warning about an unexpected event to the user. The messages coming out of this level may not halt the progress of the system.
INFO	Gives the progress and chosen state information. This level is useful for the end user.
DEBUG	Helps the developer debug the application. Level of the message logged is focused on providing support to an application developer.

Log level	Description
TRACE	Gives more detailed information than the DEBUG level.

## Events and generic webhook variables

You can customize your [generic webhook \(page 1076\)](#) response with data from Contrast events such as `NEW_VULNERABILITY` and `SERVER_OFFLINE`. Each event contains [general \(page 1078\)](#), [application \(page 1078\)](#), [server \(page 1078\)](#) or [vulnerability \(page 1078\)](#) variables you can call in your payload request.

Event	Variables
ATTACK_END	<a href="#">General (page 1078)</a> , <a href="#">Application (page 1077)</a> , <a href="#">Server (page 1078)</a>
ATTACK_EVENT_COMMENT	<a href="#">General (page 1078)</a> , <a href="#">Application (page 1078)</a> , <a href="#">Server (page 1078)</a>
ATTACK_UPDATE	<a href="#">General (page 1078)</a> , <a href="#">Application (page 1078)</a> , <a href="#">Server (page 1078)</a>
EXPIRING_LICENSE	<a href="#">General (page 1078)</a> , <a href="#">Application (page 1078)</a>
NEW_ASSET (if new application)	<a href="#">General (page 1078)</a> , <a href="#">Application (page 1078)</a> and <a href="#">Server (page 1078)</a> (if new application)
NEW_ATTACK_APPLICATION	<a href="#">General (page 1078)</a> , <a href="#">Application (page 1078)</a> , <a href="#">Server (page 1078)</a>
NEW_ATTACK_UPDATE	<a href="#">General (page 1078)</a> , <a href="#">Application (page 1078)</a> , <a href="#">Server (page 1078)</a>
NEW_ATTACK	<a href="#">General (page 1078)</a> , <a href="#">Application (page 1078)</a> , <a href="#">Server (page 1078)</a>
NEW_VULNERABILITY_COMMENT	<a href="#">General (page 1078)</a> , <a href="#">Application (page 1078)</a> , <a href="#">Server (page 1078)</a> , <a href="#">Vulnerability (page 1078)</a>
NEW_VULNERABILITY	<a href="#">General (page 1078)</a> , <a href="#">Application (page 1078)</a> , <a href="#">Server (page 1078)</a> , <a href="#">Vulnerability (page 1078)</a>
NEW_VULNERABLE_LIBRARY	<a href="#">General (page 1077)</a> , <a href="#">Application (page 1078)</a>
SERVER_OFFLINE	<a href="#">General (page 1078)</a> , <a href="#">Server (page 1078)</a>
VULNERABILITY_CHANGESTATUS_CLOSED	<a href="#">General (page 1078)</a> , <a href="#">Application (page 1078)</a> , <a href="#">Server (page 1078)</a> , <a href="#">Vulnerability (page 1078)</a>
VULNERABILITY_CHANGESTATUS_OPEN	<a href="#">General (page 1078)</a> , <a href="#">Application (page 1078)</a> , <a href="#">Server (page 1078)</a> , <a href="#">Vulnerability (page 1078)</a>
VULNERABILITY_DUPLICATE	<a href="#">General (page 1078)</a> , <a href="#">Application (page 1078)</a> , <a href="#">Server (page 1078)</a> , <a href="#">Vulnerability (page 1078)</a>

## Generic webhook variables

You can customize your [generic webhook \(page 1076\)](#) response with data from Contrast events such as `NEW_VULNERABILITY` and `SERVER_OFFLINE`. Each event contains variables you can call in your payload request. Variables are either for general use or for an application, server or vulnerability.

Variables	Description
<b>General variables</b>	
<code>\$EventType</code>	The event type responsible for triggering the webhook For example: <code>SERVER_OFFLINE</code>
<code>\$Message</code>	A message summarizing the event that triggered the webhook
<code>\$OrganizationId</code>	The unique ID Contrast assigns to an organization when it is created
<code>\$OrganizationName</code>	The name of your organization
<code>\$Title</code>	Always returns "Contrast Security"
<b>Application variables</b>	
<code>\$ApplicationChild</code>	Returns true if the application is a child application, false if not
<code>\$ApplicationCode</code>	A secondary shorthand that appears in the title of an application, and is blank by default For example: <code>TEST</code>
<code>\$ApplicationContextPath</code>	The context path of the application For example: <code>/example/somethingelse</code>

Variables	Description
\$ApplicationFirstSeen	When the application was first seen, in Unix time For example: 1572033840000
\$ApplicationHasParentApp	Returns true if the application has a parent, false if not
\$ApplicationImportance	Enumerated value of the application Importance level For example: MEDIUM
\$ApplicationId	The unique ID Contrast assigns to an application when it is created For example: 49fe2978-1833-4441-83db-2b7o486d9413
\$ApplicationImportanceDescription	The importance level assigned to the application For example: Medium
\$ApplicationLanguage	The programming language of the application
\$ApplicationLastSeen	When the application was last seen, in Unix time For example: 1572033840000
\$ApplicationLicenseLevel	Whether or not the application has an Assess license Values: Licensed, Unlicensed
\$ApplicationMaster	Returns true if the application is a primary application, false if not
\$ApplicationName	The name of the application
\$ApplicationParentAppId	The unique ID Contrast assigns to an application when it's created, in this case, the parent application, if it exists For example: 49fe2978-1833-4441-83db-2b7o486d9413
\$ApplicationTags	A comma separated list of the Application tags.
\$ApplicationTotalModules	The number of modules your application has
<b>Server variables</b>	
\$Environment	The environment of the server For example: DEVELOPMENT or PRODUCTION
\$ServerId	The ID of the server involved in the event If more than one server is involved, this is a comma-delimited list of server IDs.
\$ServerName	The name of the server involved in the event If more than one server is involved, this is a comma-delimited list of server names
<b>Vulnerability variables</b>	
\$Severity	If this event is triggered by a vulnerability, this is the severity of the vulnerability
\$Status	If this event is triggered by a vulnerability, this is the status of the vulnerability
\$TraceId	If this event is triggered by a vulnerability, this is the vulnerability ID
\$VulnerabilityAgentLanguage	The application language or framework name of the where the vulnerability was discovered (for example,.Java, .NET, Ruby, and so forth.)
\$VulnerabilityAppVersionTags	The application versions the vulnerability is found in For example: v1.2.3
\$VulnerabilityAutoRemediatedExpirationPeriod	Auto-remediated expiration period for the vulnerability, in Unix time For example: 1572033840000
\$VulnerabilityBugTrackerTickets	A comma delimited list of tickets created when the vulnerability was sent to bugtracker For example: ticket1, ticket2, ticket3
\$VulnerabilityCategory	The category of vulnerability found For example: Injection
\$VulnerabilityClosedTime	When the vulnerability was closed, in Unix time For example: 1572033840000
\$VulnerabilityConfidence	Confidence of the vulnerability

Variables	Description
\$VulnerabilityDefaultSeverity	Default severity of the vulnerability
\$VulnerabilityDiscovered	When the vulnerability was first discovered, in Unix time For example: 1572033840000
\$VulnerabilityEvidence	The evidence of the vulnerability
\$VulnerabilityInstanceUuid	The unique ID Contrast assigns to a vulnerability instance when it is created For example: R33T-N00B-TGIF-RM6P
\$VulnerabilityFirstTimeSeen	When the vulnerability was first seen, in Unix time For example: 1572033840000
\$VulnerabilityImpact	The impact level of the vulnerability Values: Low, Medium, High
\$VulnerabilityLastTimeSeen	Last time the vulnerability was seen, in Unix time For example: 1572033840000
\$VulnerabilityInstanceLastTimeSeen	Last time the vulnerability was seen, in Unix time For example: 1572033840000
\$VulnerabilityLicenseLevel	License level of the vulnerability
\$VulnerabilityLikelihood	The likelihood of the vulnerability Values: Low, Medium, High
\$VulnerabilityReportedToBugTracker	When the vulnerability was sent to a bugtracker, in Unix time For example: 1572033840000
\$VulnerabilityReportedToBugTrackerTime	Returns true If the vulnerability was sent to a bugtracker
\$VulnerabilityRule	Rule associated with the vulnerability
\$VulnerabilityRuleName	Name of the rule associated to the vulnerability
\$VulnerabilityRuleTitle	Title of the rule associated to the vulnerability
\$VulnerabilitySubStatus	Substatus of the vulnerability
\$VulnerabilityTags	Custom tags associated with the vulnerability For example: my-custom-tag
\$VulnerabilityTitle	Title of the vulnerability
\$VulnerabilitySubStatusKeyCode	Key code of the vulnerability substatus
\$VulnerabilityTotalTracesReceived	Total number of times the vulnerability was received
\$VulnerabilityUuid	The unique ID used to look up a vulnerability
\$VulnerabilityVisible	true if the vulnerability is licensed and visible, false if not
\$VulnerabilityRule	If event is triggered by a vulnerability, this is the rule that the vulnerability violated
\$VulnerabilityTags	If event is triggered by a vulnerability, this is a comma-delimited list of tags associated with the vulnerability

## Regular expression reference

Use this table, and the examples below, for reference when [creating application exclusions](#) (page 1142):

Effect	Pattern	Example pattern	Example match
Start of a string	^	^w+	Start of a string
End of a string	\$	w+\$	End of a string
Case-insensitive match of following string	(?i)	(?i)%0a	%0a or %0A
A single character of: a, b or c	[abc]	[abc]+	a bb ccc
A character except: a, b or c	[^abc]	[^abc]+	Anythingbutabc.
A character in the range: a-z	[a-z]	[a-z]+	Only a-z
A character not in the range: a-z	[^a-z]	[^a-z]+	Anythingbuta-z.
A character in the range of: a-z or A-Z	[a-zA-Z]	[a-zA-Z]+	abc123DEF
Any single character	.	.+	abc
Any whitespace character	\s	\s	anywhitespacecharacter

Effect	Pattern	Example pattern	Example match
Any non-whitespace character	\S	\S+	any non-whitespace
Any digit	\d	\d	not 1 not 2
Any non-digit	\D	\D+	not 1 not 2
Zero or one of a	a?	ba?	ba b a
Zero or more of a	a*	ba*	a ba baa aaa ba b
One or more of a	a+	a+	a aa aaa aaaa bab baab
Exactly 3 of a	a{3}	a{3}	a aa aaa aaaa
3 or more of a	a{3,}	a{3,}	a aa aaa aaaa aaaaaa
Between 3 and 6 of a	a{3,6}	a{3,6}	a aa aaa aaaa aaaaaa aaaa
Period (dot) is a literal character	.	a.b	string.string

## Supported browsers

Contrast is web-based application for HTML5, and the interface is based on React and AngularJS. It works well with the latest version of any modern browser. Contrast actively tests our product in the **current and last major version** of the following browsers:

- Chrome
- Edge
- Firefox
- Safari

Opera browsers or older versions of Internet Explorer, Firefox, or Safari browsers may still work with Contrast, but some features may not display as intended.

## Contrast Beta Terms and Conditions

Contrast beta products and features adhere to these conditions:

- This product is in active development and undergoing continuous improvements.
- These beta products are provided as-is without warranty of any kind.
- Since beta products are still in development, there may be issues. Do not use beta versions in production environments.
- Customers may be required to sign additional agreements in order to use beta products.
- The beta program is for qualified users who want early access to new features and enhancements, and who agree to provide feedback. Please send comments, questions or bug reports about beta products to [support@contrastsecurity.com](mailto:support@contrastsecurity.com).

## Privacy and data collection

It's important that Contrast Security understands how customers are using our products so they can be improved. This information helps Contrast Security resolve problems and fix bugs.

We collect data through:

- [System diagnostics \(page 1250\)](#)
- [.NET Framework and .NET Core agent telemetry \(page 269\)](#)
- [Ruby telemetry \(page 529\)](#)
- [Python telemetry \(page 467\)](#)

The collected data is anonymous and does not contain application data. It is collected by Contrast Security, and is never shared. It is governed by [Contrast Security's Privacy Policy](#).

Protecting your privacy is important to us. If you suspect we are collecting sensitive data or the data is being insecurely or inappropriately handled, send an email to [security@contrastsecurity.com](mailto:security@contrastsecurity.com) for investigation.



## Preview releases

The content in this section represents new features currently not publicly available on the Contrast platform. These features are part of the Contrast pre-release customer testing program. If you want to be included in one or more of these areas, contact your Contrast representative.

### Topics in this section

- [Role-based access control \(page 1277\)](#)
- [Visual Studio Code \(page 1320\)](#)

### Role-based access control (Preview) ☁

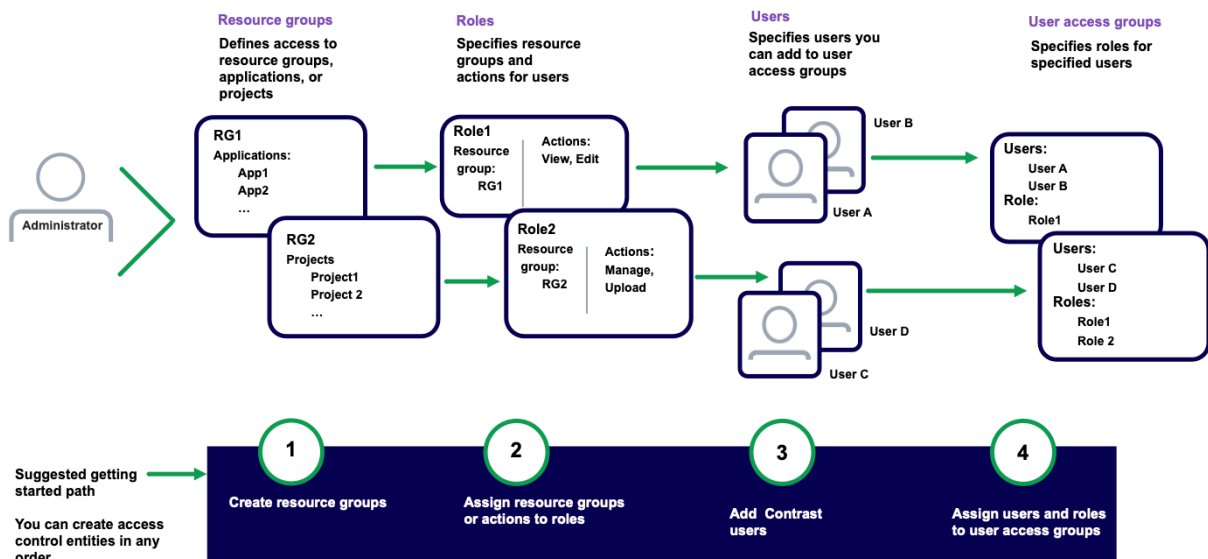
To manage user access to applications, projects, and organization settings in Contrast, set up resource groups, roles, and user access groups.



#### NOTE

This feature is supported for hosted customers only

On-premises customers manage Contrast access by setting up [organization users and access groups \(page 1163\)](#).



### Access control settings

Access control requires these settings:

- **Resource groups:** (page 1284) These groups determine which applications, SCA project groups, and scan projects users can access, depending on their assigned role. You can use built-in resource groups or create custom groups.
- **Roles:** (page 1291) Roles define the actions that users have permissions to perform. You can use built-in roles or create custom roles.

The built-in role, Organization administrator, lets you manage all user access groups, roles, and resource groups.

- **Users: (page 1300)** The list of Contrast users who you can add to user access groups.
- **User access groups: (page 1289)** These groups assign roles to users.  
You can use built-in user access groups or create custom user access groups.

## Methods for managing access control

To manage access control, use either of these methods:

- **Contrast web interface**  
The Contrast web interface provides a visual method for setting up and managing access control. This method requires an Organization Administrator or Organization Admin role.
- **Access control APIs**  
The access control APIs are useful if you want to automate access control management or integrate it into existing systems. This method requires the `PLATFORM_ORG_MANAGE` permission.  
Read more about the [access control APIs](#).

## Naming standards and requirements

Names for resource groups, roles, and user access groups must be unique for each organization. For example, you can't create two roles with the name, **MyAdmins**.

Before you create custom roles or groups, consider developing a naming standard for all custom settings. Doing so will help make managing a large number of custom settings easier. One approach is to map the names of resource groups to the roles that you would associate with those groups. For example, if you create a resource group named **Ecommerce**, a good role name would be **Ecommerce Developer**. The users with this role are your developers working on ecommerce applications.

### Naming requirements for all custom settings

- Names can contain up to 50 alphanumeric characters.
- Descriptions can contain up to 1,024 characters.
- You can use special characters and spaces in names and descriptions.
- Names are case insensitive.  
For example, ecommerce developers and Ecommerce Developers are considered to be the same name.

## Access Control APIs

The [Access Control APIs](#) let you do many of the role-based access control tasks that you can do from the Contrast web interface.

## Actions and permissions (Preview) ☹

Each action that you assign to a role provides permissions to perform specific tasks and access to data.

[Actions can inherit permissions from other actions. \(page 1284\)](#) For example, the Edit application action includes all of the permissions that the View application action provides. In this case, only need to assign the Edit application action to a role and not both the Edit application and View application actions.

**NOTE**

This feature is supported for hosted customers only and is in preview mode. For access to this feature, contact [Contrast support](#).

On-premises customers manage Contrast access by setting up [organization users and access groups \(page 1163\)](#).

## Organization actions and permissions

This action:	Includes these permissions:	And is part of these built-in roles
View organization	<b>General</b> <ul style="list-style-type: none"><li>• Log in to the Contrast web interface</li><li>• View your user profile</li><li>• View notifications</li></ul> <b>Reports</b> <ul style="list-style-type: none"><li>• Generate application reports</li></ul>	Organization viewer Organization editor Organization administrator Organization rules administrator App security administrator DevOps administrator
Edit organization	<b>General</b> <ul style="list-style-type: none"><li>• View your user profile</li><li>• View notifications</li><li>• View score settings</li><li>• View report settings</li><li>• View organization settings</li><li>• Access the Add new button</li></ul> <b>Policies</b> <ul style="list-style-type: none"><li>• View policies</li></ul> <b>Security</b> <ul style="list-style-type: none"><li>• Get agent keys</li><li>• Manage agent keys</li></ul> <b>Reports</b> <ul style="list-style-type: none"><li>• View report settings</li><li>• Generate application reports</li></ul>	Organization editor Organization administrator Organization rules administrator
Manage organization rules	<b>General</b> <ul style="list-style-type: none"><li>• View your user profile</li><li>• View notifications</li><li>• View score settings</li><li>• View organization settings</li><li>• Enable or disable Protect</li></ul> <b>Policies</b> <ul style="list-style-type: none"><li>• Manage scoring policies</li></ul> <b>Reports</b> <ul style="list-style-type: none"><li>• View report settings</li><li>• Generate application reports</li></ul>	Organization administrator Organization rules administrator

This action:	Includes these permissions:	And is part of these built-in roles										
Manage organization	<table><tr><td><b>General</b><ul style="list-style-type: none"><li>• View your user profile</li><li>• View notifications</li><li>• View organization settings</li></ul></td><td><b>Integrations</b><ul style="list-style-type: none"><li>• View integration details</li><li>• Edit integration settings</li></ul></td></tr><tr><td><b>Servers</b><ul style="list-style-type: none"><li>• Manage server licenses</li></ul></td><td><b>Users</b><ul style="list-style-type: none"><li>• View users</li><li>• Create users</li><li>• Delete users</li><li>• Edit user settings</li></ul></td></tr><tr><td><b>Policies</b><ul style="list-style-type: none"><li>• Manage scoring policies</li></ul></td><td><b>Reports</b><ul style="list-style-type: none"><li>• View report settings</li><li>• Change report settings</li><li>• Generate application reports</li></ul></td></tr><tr><td><b>Security</b><ul style="list-style-type: none"><li>• Rotate API keys</li><li>• Manage IP restrictions</li><li>• Manage password policy</li><li>• Manage session timeouts</li><li>• Manage two-step verification and SAML</li></ul></td><td><b>Protect</b><ul style="list-style-type: none"><li>• Enable and disable Protect</li></ul></td></tr><tr><td><b>Scan</b><ul style="list-style-type: none"><li>• Turn on dynamic scoring</li></ul></td><td></td></tr></table>	<b>General</b> <ul style="list-style-type: none"><li>• View your user profile</li><li>• View notifications</li><li>• View organization settings</li></ul>	<b>Integrations</b> <ul style="list-style-type: none"><li>• View integration details</li><li>• Edit integration settings</li></ul>	<b>Servers</b> <ul style="list-style-type: none"><li>• Manage server licenses</li></ul>	<b>Users</b> <ul style="list-style-type: none"><li>• View users</li><li>• Create users</li><li>• Delete users</li><li>• Edit user settings</li></ul>	<b>Policies</b> <ul style="list-style-type: none"><li>• Manage scoring policies</li></ul>	<b>Reports</b> <ul style="list-style-type: none"><li>• View report settings</li><li>• Change report settings</li><li>• Generate application reports</li></ul>	<b>Security</b> <ul style="list-style-type: none"><li>• Rotate API keys</li><li>• Manage IP restrictions</li><li>• Manage password policy</li><li>• Manage session timeouts</li><li>• Manage two-step verification and SAML</li></ul>	<b>Protect</b> <ul style="list-style-type: none"><li>• Enable and disable Protect</li></ul>	<b>Scan</b> <ul style="list-style-type: none"><li>• Turn on dynamic scoring</li></ul>		Organization administrator
<b>General</b> <ul style="list-style-type: none"><li>• View your user profile</li><li>• View notifications</li><li>• View organization settings</li></ul>	<b>Integrations</b> <ul style="list-style-type: none"><li>• View integration details</li><li>• Edit integration settings</li></ul>											
<b>Servers</b> <ul style="list-style-type: none"><li>• Manage server licenses</li></ul>	<b>Users</b> <ul style="list-style-type: none"><li>• View users</li><li>• Create users</li><li>• Delete users</li><li>• Edit user settings</li></ul>											
<b>Policies</b> <ul style="list-style-type: none"><li>• Manage scoring policies</li></ul>	<b>Reports</b> <ul style="list-style-type: none"><li>• View report settings</li><li>• Change report settings</li><li>• Generate application reports</li></ul>											
<b>Security</b> <ul style="list-style-type: none"><li>• Rotate API keys</li><li>• Manage IP restrictions</li><li>• Manage password policy</li><li>• Manage session timeouts</li><li>• Manage two-step verification and SAML</li></ul>	<b>Protect</b> <ul style="list-style-type: none"><li>• Enable and disable Protect</li></ul>											
<b>Scan</b> <ul style="list-style-type: none"><li>• Turn on dynamic scoring</li></ul>												
Manage platform organization	<b>Access control</b> <ul style="list-style-type: none"><li>• Manage user access groups</li><li>• Manage roles</li><li>• Manage resource groups.</li></ul> <b>Reports</b> <ul style="list-style-type: none"><li>• Generate application reports</li></ul>	Organization administrator DevOps administrator										
View audit logs	<b>Security</b> <ul style="list-style-type: none"><li>• View audit log details</li></ul>	Organization administrator										

## Application actions and permissions

This action:	Includes these permissions:		And is part of these built-in roles
View application	<b>General</b>	<b>Vulnerabilities</b>	Application viewer
	<ul style="list-style-type: none"><li>• View your user profile</li><li>• View notifications</li><li>• Customize score settings</li><li>• Change report settings</li><li>• Change notification settings</li></ul>	<ul style="list-style-type: none"><li>• View vulnerability details</li><li>• Export vulnerabilities, traces, and routes</li><li>• Replay HTTP requests</li><li>• Manage discussion</li></ul>	Application editor
	<b>Applications</b>	<b>Policies</b>	Application administrator
	<ul style="list-style-type: none"><li>• View application details</li><li>• View, edit, and delete tags</li></ul>	<ul style="list-style-type: none"><li>• View application policies</li><li>• View application exclusions</li></ul>	Application rules administrator
	<b>Servers</b>	<b>Reports</b>	App security administrator
	<ul style="list-style-type: none"><li>• View server details</li></ul>	<ul style="list-style-type: none"><li>• Generate application reports</li></ul>	App security engineer
	<b>Libraries</b>		DevOps administrator
	<ul style="list-style-type: none"><li>• View library details</li><li>• View tags</li><li>• View manifest details</li><li>• Export libraries</li></ul>		

This action:	Includes these permissions:		And is part of these built-in roles
Edit application	<b>General</b> <ul style="list-style-type: none"> <li>• View your user profile</li> <li>• View notifications</li> <li>• View organization settings</li> <li>• View score settings</li> <li>• Change notification settings</li> </ul>	<b>Vulnerabilities</b> <ul style="list-style-type: none"> <li>• View vulnerability details</li> <li>• Send vulnerabilities to a bug tracker</li> <li>• Merge vulnerabilities</li> <li>• Edit vulnerability settings</li> <li>• Manage discussions</li> <li>• Delete vulnerabilities</li> <li>• Export vulnerabilities, traces, and routes</li> <li>• Replay HTTP requests</li> <li>• Manage traces</li> </ul>	Application editor Application administrator Application rules administrator App security administrator
	<b>Applications</b> <ul style="list-style-type: none"> <li>• View application details</li> <li>• View tags</li> <li>• Merge applications</li> <li>• Archive applications</li> <li>• Restore applications</li> <li>• Edit and delete tags</li> <li>• Manage filters</li> </ul>	<b>Policies</b> <ul style="list-style-type: none"> <li>• View application policies</li> <li>• View application exclusions</li> <li>• Manage scoring policies</li> </ul>	
	<b>Servers</b> <ul style="list-style-type: none"> <li>• View server details</li> <li>• Edit and delete tags</li> <li>• Edit server settings</li> </ul>	<b>Reports</b> <ul style="list-style-type: none"> <li>• Generate application reports</li> <li>• Change report settings</li> </ul>	
	<b>Libraries</b> <ul style="list-style-type: none"> <li>• View library details</li> <li>• View tags</li> <li>• Edit and delete tags</li> <li>• View manifest details</li> <li>• Export libraries</li> </ul>	<b>Protect</b> <ul style="list-style-type: none"> <li>• Enable and disable Protect</li> </ul>	
Manage application rules	<b>General</b> <ul style="list-style-type: none"> <li>• View your user profile</li> <li>• View notifications</li> </ul> <b>Architecture</b> <ul style="list-style-type: none"> <li>• View architecture details</li> </ul> <b>Policies</b> <ul style="list-style-type: none"> <li>• Manage Assess rule settings</li> <li>• Manage library policies</li> <li>• Manage remediation policies</li> </ul>	<b>Protect</b> <ul style="list-style-type: none"> <li>• Enable and disable Protect</li> </ul>	Application administrator Application rules administrator App Security administrator

This action:	Includes these permissions:		And is part of these built-in roles
Manage application	<b>General</b> <ul style="list-style-type: none"> <li>Log in to the Contrast web interface</li> <li>View your user profile</li> <li>View notifications</li> <li>View organization settings</li> <li>View score settings</li> <li>Change report settings</li> <li>Change notification settings</li> </ul> <b>Applications</b> <ul style="list-style-type: none"> <li>View application details</li> <li>View tags</li> <li>Edit and delete tags</li> <li>Merge applications</li> <li>Archive applications</li> <li>Restore applications</li> <li>Manage filters</li> <li>Manage traces</li> <li>Manage bug tracker settings</li> <li>License applications</li> <li>Edit application settings</li> <li>Reset applications</li> <li>Delete applications</li> </ul> <b>Servers</b> <ul style="list-style-type: none"> <li>View server details</li> <li>Edit and delete tags</li> <li>Edit server settings</li> <li>Delete servers</li> <li>Manage server licenses</li> </ul> <b>Integrations</b> <ul style="list-style-type: none"> <li>View integration details</li> <li>Edit integration settings</li> </ul> <b>Libraries</b> <ul style="list-style-type: none"> <li>View library details</li> <li>View tags</li> <li>Edit and delete tags</li> <li>View manifest details</li> <li>Export libraries</li> </ul> <b>Vulnerabilities</b> <ul style="list-style-type: none"> <li>View vulnerability details</li> <li>Send vulnerabilities to a bug tracker</li> <li>Merge vulnerabilities</li> <li>Edit vulnerability settings</li> <li>Manage discussions</li> <li>Delete vulnerabilities</li> <li>Export vulnerabilities, traces, and routes</li> <li>Replay HTTP requests</li> <li>Manage traces</li> </ul>	<b>Policies</b> <ul style="list-style-type: none"> <li>View applications policies</li> <li>View application exclusions</li> <li>View scoring policies</li> </ul> <b>Architecture</b> <ul style="list-style-type: none"> <li>View architecture details.</li> </ul> <b>Protect</b> <ul style="list-style-type: none"> <li>Enable and disable Protect</li> </ul> <b>Reports</b> <ul style="list-style-type: none"> <li>Generate application reports</li> <li>View report settings</li> </ul> <b>Users</b> <ul style="list-style-type: none"> <li>Manage resource groups</li> </ul>	Application administrator

## Project actions and permissions

This action:	Includes these permissions:	And is part of these built-in roles
View project	<b>Scan</b> <ul style="list-style-type: none"> <li>View details about scan projects and scans</li> <li>Download scan results in a SARIF or CSV file</li> </ul>	Project viewer Project administrator App Security administrator DevOps administrator
Upload scans	<b>Scan</b> <ul style="list-style-type: none"> <li>Upload files for scanning</li> </ul>	Scan uploader Project administrator App security administrator

This action:	Includes these permissions:	And is part of these built-in roles
View, edit, delete project	<b>Scan</b> <ul style="list-style-type: none"> <li>• Archive scan projects</li> <li>• Delete scan projects</li> <li>• Edit project settings</li> <li>• Change scan vulnerability status and severity</li> <li>• Upload files for scanning</li> <li>• Start scans</li> <li>• View details for scan projects and scans</li> <li>• Download scan results in a SARIF or CSV file</li> </ul>	Project administrator App security administrator
Create project	<b>Scan</b> <ul style="list-style-type: none"> <li>• Create scan projects</li> </ul>	Project administrator
Edit project	<b>Scan</b> <ul style="list-style-type: none"> <li>• Edit scan project settings</li> <li>• Upload files for scanning</li> <li>• View details for scan projects and scans</li> <li>• Download scan results in a SARIF or CSV file</li> </ul>	Project administrator App security administrator
Delete project	<b>Scan</b> <ul style="list-style-type: none"> <li>• View details for scan projects and scans</li> <li>• Download scan results in a SARIF or CSV file</li> <li>• Delete scan projects</li> </ul>	Project administrator App security administrator

## Protect actions and permissions

This action:	Includes these permissions:	And is part of these built-in resource groups
Access Protect	<b>Protect</b> <ul style="list-style-type: none"> <li>• View Protect data</li> </ul>	Protect viewer Protect policy administrator Organization administrator
Manage Protect exclusions	<b>Protect</b> <ul style="list-style-type: none"> <li>• View application exclusions for Protect rules</li> <li>• Create and edit application exclusions for Protect rules</li> </ul>	Protect exclusion administrator Organization administrator
Manage Protect policies	<b>Protect</b> <ul style="list-style-type: none"> <li>• View and edit Protect policies</li> <li>• Add or edit log enhancers</li> <li>• Manage attack alerts</li> <li>• Manage the IP denylist</li> <li>• Manage Protect licenses</li> <li>• Manage Protect rule settings.</li> </ul>	Protect policy administrator
Manage sensitive data policies	<b>Protect</b> <ul style="list-style-type: none"> <li>• View details for sensitive data policies</li> <li>• Edit policies for sensitive data</li> </ul>	Protect sensitive data administrator Organization administrator
View attack data	<b>Protect</b> <ul style="list-style-type: none"> <li>• View attack event data</li> </ul>	App security engineer

## SCA projects actions and permissions

This action:	Includes these permissions:	And is part of these built-in roles
View SCA projects	<b>Libraries</b> <ul style="list-style-type: none"> <li>View SCA project details</li> </ul>	SCA project group viewer SCA project group administrator

This action:	Includes these permissions:	And is part of these built-in roles
Create SCA projects	<b>Libraries</b> <ul style="list-style-type: none"><li>• Connect to open source repositories for new SCA projects</li><li>• Create (and track, if using the <code>--track</code> option) new SCA project from the Contrast CLI</li></ul>	SCA project group administrator
Delete SCA projects	<b>Libraries</b> <ul style="list-style-type: none"><li>• Disconnect open source repositories for SCA projects</li></ul>	SCA project group administrator
Manage SCA projects	<b>Libraries</b> <ul style="list-style-type: none"><li>• View SCA project details</li><li>• Connect to open source repositories for SCA projects</li><li>• Edit SCA project details</li><li>• Disconnect SCA projects</li><li>• Run additional or subsequent SCA scans for SCA projects</li></ul>	SCA project group administrator

## Serverless actions and permissions

This action:	Includes these permissions:	And is part of these built-in roles
View Serverless	<b>Serverless</b> <ul style="list-style-type: none"><li>• View Serverless data</li><li>• Interact with API endpoints</li></ul>	Serverless user

## Inherited actions

This action:	Inherits permissions from this action:

## Resource groups (Preview)

Resource groups let you specify the applications, projects, and organization settings that users can access, based on their assigned roles.



### NOTE

This feature is supported for hosted customers only and is in preview mode. For access to this feature, contact [Contrast support](#).

On-premises customers manage access to Contrast capabilities by setting up [organization users and access groups \(page 1163\)](#).

## Resource groups tab

The Resource groups tab displays the list of existing groups. From this tab, you can:

- View a list of resource groups.  
Use search to find specific groups.
- Manage resource groups:



- [Add a resource group \(page 1287\)](#).
- [Edit a resource group \(page 1285\)](#).
- [Delete a resource group \(page 1286\)](#).

**Organization Settings**

Organization  
Groups  
Users  
Access Control  
Security  
Agent  
Single Sign-On  
Integrations  
Servers  
Applications  
Notifications  
Report Settings  
Score Settings

Users **Resource Groups** Roles User Access Groups

**Resource Groups**

Search

+ Add Group

RESOURCE GROUP		DESCRIPTION	
All applications	Built-in	All applications added to Contrast. Provides access to Assess and Protect data. Actions define the permissions a user has for this data.	
All projects	Built-in	All scan projects defined in the system. Actions define the permissions a user has for projects.	
All roles	Built-in	Includes all built-in and custom roles. Actions define the permissions a user has for roles.	
All access control settings	Built-in	Includes all settings for access control entities. Actions define the permissions a user has for the entities.	
Ecommerce projects		Projects for Ecommerce development	

## Built-in resource groups

You can select these built-in resource groups for access control roles:

- **All applications:** Provides access to all applications in your organization.
- **All Protect exclusions:** Provides access to all Protect exclusion settings
- **All Protect sensitive data policies:** Provides access to all sensitive-date policy settings.
- **All functions:** Provides access to all Serverless functions and data endpoints.
- **All projects:** Provides access to all scan projects in your organization.
- **All roles:** Provides access to all roles in your organization only.
- **All access control settings:** Provides access to all settings for users, roles, user access groups, and resource groups.
- **All organization settings:** Provides access to all organization settings, including management of all users access groups, resource groups, and roles.
- **All user access groups:** Provides access to all user access groups.
- **All resource groups:** Provides access to all resource groups.
- **All SCA project groups:** Provides access to all SCA projects.



### NOTE

You cannot change the settings for built-in resource groups. To view the settings for these groups, select the **View** icon ().

## Edit custom resource groups (Preview)

Use this procedure to change settings for a custom resource group. You cannot change settings for built-in resource groups.

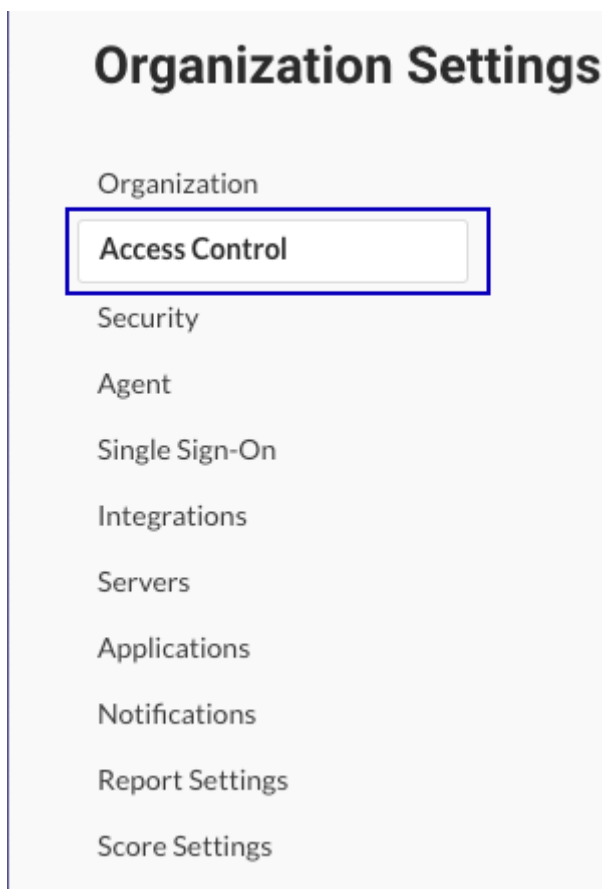
**NOTE**

This feature is supported for hosted customers only and is in preview mode. For access to this feature, contact [Contrast support](#).

On-premises customers manage access to Contrast by setting up [organization users and access groups \(page 1163\)](#).

**Steps**

1. From the user menu, select **Organization settings**.
2. Select **Access control**.



3. Select the **Resource groups** tab.
4. Select the **Edit** icon (✎) at the end of the row for the group you want to change.
5. Change the settings for the resource group and select **Save**.

**Delete custom resource groups (Preview)** ☹

Use this procedure to delete a custom resource group. You cannot delete built-in resource groups.

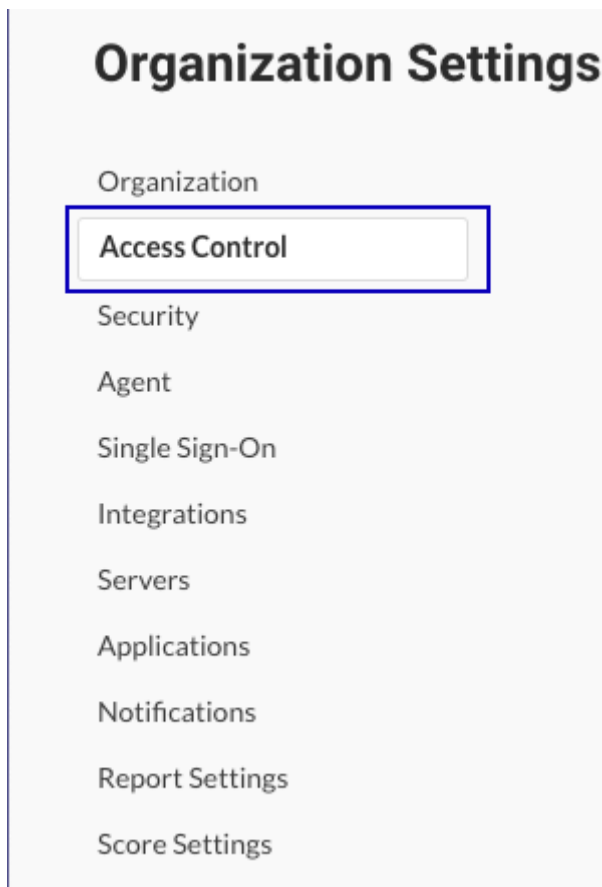
**NOTE**

This procedure is supported for hosted customers only and is in preview mode. For access to this feature, contact [Contrast support](#).

On-premises customers manage access to Contrast by setting up [organization users and access groups \(page 1163\)](#).

**Steps**

1. From the user menu, select **Organization settings**.
2. Select **Access control**.



3. Select the **Resource groups** tab.
4. Select the **Delete** icon (🗑️) at the end of the row for the group you want to delete.
5. In the Delete resource groups window, select **Delete**.

**Add resource groups (Preview)** ☹️

Custom resource groups let you specify the applications, projects, and organization settings that users can access.

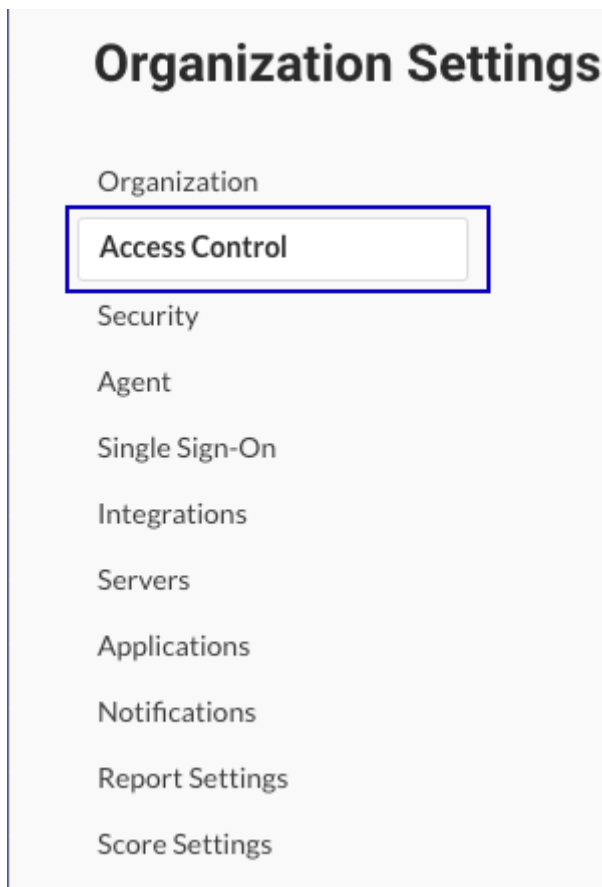
**NOTE**

This feature is supported for hosted customers only and is in preview mode. For access to this feature, contact [Contrast support](#).

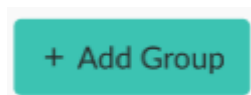
On-premises customers manage Contrast access by setting up [organization users and access groups \(page 1163\)](#).

**Steps**

1. From the user menu, select **Organization settings**.
2. Select **Access control**.



3. Select the **Resource groups** tab.
4. Select **Add group**.



5. In the Add resource group window, specify these settings:

- **Resource group name:** Specify a descriptive name.  
The name must be unique for the organization. Use up to 255 characters, including spaces and special characters.
- **Resource group description:** Specify a description that's easy to understand.  
Consider using a description that identifies the purpose of the group. Use up to 1,024 characters, including spaces and special characters.
- **Resource groups:** Select resource groups to include in this group.  
The list of available resource groups might not show all individual resource groups that exist in your organization. If an individual resource group is part of another resource group, you can't select the individual group. You can, however, select the resource group that includes the individual resource group. For example:
  - You create Resource Group 1 and include MyResource group.
  - When you create Resource Group 2, you don't see MyResource group in the list. You do see Resource Group 1, which you can add to Resource Group 2.
 If you select resource groups, selecting an individual project or application is not allowed. You might want to nest resource groups as a way of providing easy access to multiple resource groups for specific users. For example, if you have multiple resource groups for your Ecommerce development, and some users need access to all of these groups, you could create a main Ecommerce resource group that includes all the lower level Ecommerce resource groups
- **Scan projects:** Select individual scan projects to include in this group.  
If you select individual scan projects, selecting resource groups is not allowed.
- **Applications:** Select individual applications to include in this group.  
If you select individual applications, selecting resource groups is not allowed.
- **SCA project groups:** Select individual SCA project groups to include in this group.  
If you select individual project groups, selecting resource groups is not allowed.

## 6. Select **Add**.

You can now assign this resource group to a role.

## User access groups (Preview)

User access groups let you specify specific roles and resource groups for one or more users. Roles and resource groups define the permissions for users.

You can add users to built-in user access groups or create [custom groups \(page 1310\)](#).

**NOTE**

This feature is supported for hosted customers only and is in preview mode. For access to this feature, contact [Contrast support](#).

On-premises customers manage access to Contrast by setting up [organization users and access groups \(page 1163\)](#).

**User access groups tab**

The User access groups tab displays the list of existing groups. From this tab, you can:

- View a list of user access groups.  
Use search to find specific groups.
- [Add a user access group \(page 1310\)](#)
- [Edit a user access group \(page 1312\)](#)
- [Delete a user access group \(page 1313\)](#)

**Organization Settings**

Organization  
Groups  
Users  
Access Control  
Security  
Agent  
Single Sign-On  
Integrations  
Servers  
Applications  
Notifications  
Report Settings  
Score Settings

Users   Resource Groups   Roles   **User Access Groups**

**User Access Groups**

Q Search + Add Group

GROUP		DESCRIPTION	MEMBERS	
DevOps Administrators	Built-in	Users who manage application settings and projects.	0	
AppSecurity Administrators	Built-in	Users who manage applications and projects, as well as view organization data.	0	
Organization viewers	Built-in	Users who view organization data.	0	
Organization editors	Built-in	Users who view and edit organization data, such as view agent keys, manage notifications, and manage score settings.	0	
Ecommerce		Ecommerce team	1	

**Built-in user access groups**

You can add users to these built-in user access groups:

This built-in group:	Includes these built-in roles:
Organization viewers	Organization viewer
Organization editors	Organization editor
Organization rules administrators	Organization rules administrator
Organization administrators	Application administrator
	Project administrator
	Organization administrator
	Protect viewer
	Serverless user
	SCA project group administrator

This built-in group:	Includes these built-in roles:
AppSecurity administrators	App Security administrator
DevOps administrators	DevOps administrator
Developers	Organization viewer
	Protect viewer
	Serverless user
	Scan uploader
	Application viewer
	Project viewer
	SCA project group viewer
Scan manager	Project viewer
Protect viewers	Protect viewer
Serverless users	Serverless user

**NOTE**

Other than adding or removing users, you cannot change the settings for the built-in user access groups.

**See also**

[Built-in roles \(page 1291\)](#)

**Roles (Preview)** ☹

Roles let you define the applications, projects, and organization settings that users with a specific role can access.

**NOTE**

This feature is supported for hosted customers only and is in preview mode. For access to this feature, contact Contrast support.

On-premises customers manage access to Contrast by setting up [organization users and groups \(page 1163\)](#).

Contrast provides a set of built-in roles or you can add custom roles.

**Roles tab**

The Roles tab displays the list of existing roles. From this tab, you can:

- View a list of roles.  
Use search to find specific roles.
- [Add a custom role \(page 1295\)](#)
- [Edit a custom role \(page 1297\)](#).
- [Copy roles \(page 1298\)](#)

- [Delete a custom role. \(page 1299\)](#)

### Organization Settings

Organization

Access Control

Security

Agent

Single Sign-On

Integrations

Servers

Applications

Notifications

Report Settings

Score Settings

Resource Groups

Roles

User Access Groups

### Roles

Search

+ Add Role

ROLE	DESCRIPTION	
Organization editor	Organization Editor	
Organization rules administrator	Organization Rules Administrator	
Manage project	System Role to Manage All Projects	
Ecommerce developers	Developers on the Ecommerce team in NYC	<div><div></div><div></div></div>

## Built-in roles and actions

Each action associated with a role provides [permissions for a specific set of tasks and data \(page 1278\)](#).



### NOTE

You cannot change the settings for the built-in roles. To view the settings for roles, select the **View** icon (👁).

## Organization roles

This role:	Includes these built-in resource groups:	And these actions:
Organization viewer	All organization settings	View organization
Organization editor	All organization settings	Edit organization settings
Organization administrator	All access control settings	Manage organization
	All organization settings	Manage platform organization
		View audit logs
Organization rules administrator	All organization settings	Manage organization rules

## App Security roles

This role:	Includes these built-in resource groups:	And these actions:
App Security administrator	All applications	Manage application rule
	All organization settings	View, edit, delete projects
	All projects	View organization



## DevOps roles

This role:	Includes the built-in resource groups:	And these actions:
DevOps administrator	All applications	Manage organization
	All organization settings	View application
	All projects	View project
	All resource groups	View organization
	All roles	
	All user access groups	

## Application roles

This role	Includes these built-in resource groups:	And these actions:
Application viewer	All applications	View application
Application editor	All applications	Edit application
Application administrator	All applications	Manage application
Application rules administrator	All applications	Manage application rule

## Scan project roles

This role:	Includes these built-in resource groups:	And these actions:
Project viewer	All projects	View project
Scan uploader	All projects	Upload scans
Project administrator	All projects	View, edit, delete projects
		Create project

## Protect roles

This role:	Includes these built-in resource groups:	And these actions:
Protect viewer	All applications	Access Protect
Protect policies administrator	All applications	Manage Protect policies
Protect exclusions administrator	All Protect exclusions	Manage Protect exclusions
Protect sensitive data administrator	All Protect sensitive data policies	Manage protect sensitive data policies

## SCA roles

This role:	Includes these built-in resource groups:	And these actions:
SCA project group administrator	All organization settings	Create SCA projects
	All SCA project groups	View, edit, delete SCA projects
SCA project group viewer	All SCA project groups	View SCA projects

## Serverless roles

This role:	Includes these built-in resource groups:	And these actions:
Serverless user	All functions	View Serverless

## Best practices for custom roles (Preview)

If you have multiple teams collaborating on one or more projects, consider creating custom roles. Custom roles ensure that your teams have access to the appropriate resources and the correct permissions for those resources.

You can view the settings for the built-in roles to help you determine the types of actions and resource groups to add to your custom roles.

## Custom role planning

Your first step is to create a plan for your custom roles.

- **Think about the roles that people currently have.**  
For example, do you have managers and lead developers who need different access than other developers?
- **Think about the resources your teams need to work on.**  
For example, will any of your teams use static scanning to check for vulnerabilities? In this case, they will need access to scan projects.  
Consider which applications your teams are working on so you can ensure they have access to them.
- **Think about the tasks or actions that individuals need to perform for the resources they need to access.**  
For example, do individuals need to change organization settings? Do they need to be able to upload artifacts for scanning? Do they need to access application data?

## Resource groups

Consider these best practices:

- For ease in management, create resource groups that include one type of resource only. For example, keep your project resources in one group and your applications in another.  
If a built-in resource group meets your needs, you could also use it instead of creating a custom group.
- Use roles to specify the actions users can take for the different resource groups.

## Example

This example shows one approach for creating custom roles.

In this example, an administrator is creating custom roles for development teams collaborating on an ecommerce product.

### Step 1: Create a plan for all the custom roles

- **Current roles:** For this project, there are development managers, development leads, front-end developers working on a user interface, and back-end developers creating services and APIs. All of these teams collaborate together.
- **Resources:** The Ecommerce product includes shared application components that are instrumented with Contrast agents as well as scan projects for code that is scanned early in the development cycle.
- **Actions:** The development managers and leads need to be able to manage all settings for applications and scan projects. The front-end developers need to be able to access shared user interface modules. The back-end developers need to be able to access APIs and back-end services.

### Step 2: Create custom resource groups

To accommodate the different permissions you need to assign to different users, you create these resource groups:

- **Shared UI applications:** This resource group includes the UI-focused applications that front-end developers need to test at runtime.
- **Shared UI projects:** This resource group includes the UI-focused scan projects that front-end developers run to find vulnerabilities during the early stages of development.
- **Shared API applications:** This resource group includes the shared API applications that back-end developers need to test at runtime.

- **Shared back-end services:** This resource group includes the shared services that back-end developers need to test at runtime.

You group all the resource groups into a parent resource group called **Ecommerce development**.

All custom roles will include the parent group. You use actions to determine which role can access specific resources and the permissions users have for these resources.

### Step 3: Create custom roles.

You set up these roles to use the parent resource group, Ecommerce development, with specific actions:

- **Ecommerce administrator:** This role is for the Ecommerce development managers and leads. It includes these actions:
  - **Manage application rules:** Lets users perform tasks such as changing Assess or Protect rules, setting remediation policies, and setting library policies.
  - **Manage application:** Lets users change application settings.
  - **Create project:** Lets users create scan projects.
  - **View, edit, delete project:** Lets user view, edit or delete scan projects.
  - **Delete project:** Lets user delete scan projects.
  - **View organization:** Lets users perform tasks such as viewing score settings and getting API keys.
- **Ecommerce front-end developer:** This role is for the Ecommerce front-end developers. It includes these actions:
  - **View application:** Lets users view application data such as vulnerabilities, application details, and library details.
  - **Edit application:** Lets users perform tasks such as merging applications, sending data to bug trackers, editing vulnerability settings
  - **View organization:** Lets users perform tasks such as viewing score settings and getting API keys.
  - **Upload scans:** Lets users scan code for vulnerabilities for existing projects.  
The action applies to uploading code to the Contrast web interface, using the CLI, and using the Scan local engine. The user who creates projects needs the Create, View, create, edit project action.
- **Ecommerce back-end developers**
  - **View applications:** Lets users view application data such as vulnerabilities, application details, and library details.
  - **Edit application:** Lets users perform tasks such as merging applications, sending data to bug trackers, editing vulnerability settings
  - **View organization:** Lets users perform tasks such as viewing score settings and getting API keys.

### Step 4:, Create custom users groups

You create user access groups for each role and assign individual users, based on the resources they need to access.

- **Ecommerce administrators:** Includes all users with the Ecommerce administrator role.
- **Ecommerce front-end developers:** Includes all users with the Ecommerce front-end developer role.
- **Ecommerce back-end developers:** Includes all users with the Ecommerce back-end developer role.

## Add roles (Preview) ☹

Create roles to customize which Contrast resources and actions users can access.

**NOTE**

This feature is supported for hosted customers only and is in preview mode. For access to this feature, contact [Contrast support](#).

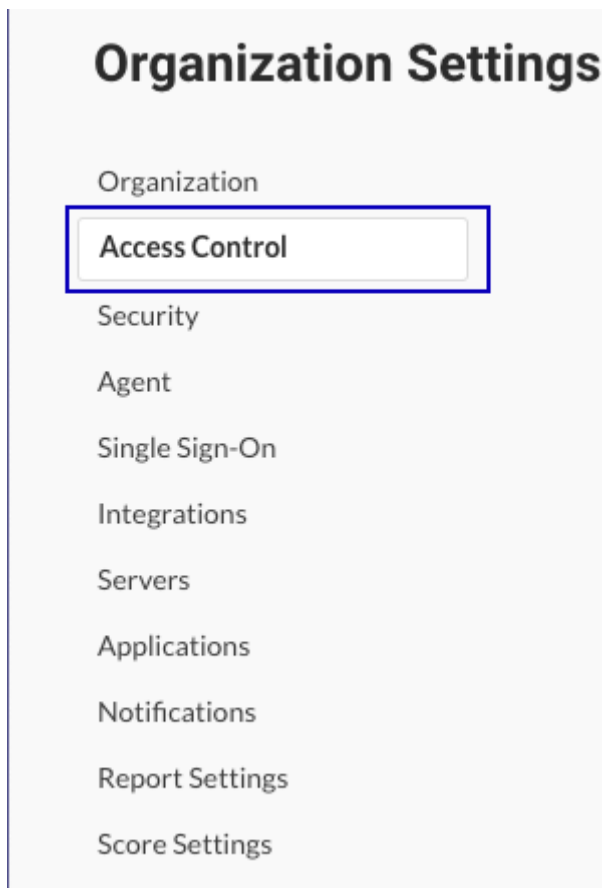
On-premises customers manage Contrast access by setting up [organization users and access groups](#) (page 1163).

**Before you begin**

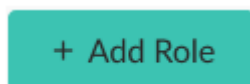
- **Hosted customers:** You need a role with the Manage user access action.
- **On-premises customers:** Manage access to Contrast by setting up [organization users and access groups](#). (page 1163)
- Determine if you need to [add a custom resource group](#) (page 1287) for the role.

**Steps**

1. From the user menu, select **Organization settings**.
2. Select **Access control**.



3. Select the **Roles** tab.
4. Select **Add role**.



5. Specify the role settings and select **Add**:

**Organization Settings**

Resource Groups **Roles** User Access Groups

### Add role

Roles define the applications and scan projects that users can access as well as permissions for specific actions.

Role name <sup>+</sup>  
Ecommerce developer

Role description <sup>+</sup>  
Developers on the Ecommerce team

**Actions** determine the permissions assigned to this role.  
**Resource groups** determine which applications and scan projects a user with this role can access.

Actions <sup>+</sup>  
SAST\_SCAN\_UPLOAD x

Resource groups <sup>+</sup>  
All Ecommerce projects x

**Your role summary** ⓘ

Resource Group Name  
Ecommerce developer

Resource Group Description  
Developers on the Ecommerce team

Resource Groups  
All Ecommerce projects

Actions  
Upload scans  
Manage project

Cancel Add

- Role name:** Specify the name for the role.  
The name must be unique for the organization. Use up to 255 characters, including spaces and special characters.
- Role description:** Specify a description of the role.  
Consider using a description that indicates the purpose of the role. Use up to 1,024 characters, including spaces and special characters.
- Actions:** Select the [actions \(page 1278\)](#) for the group.  
The list of actions is organized according to the type of resource they affect. If the actions you select don't apply to the resource groups you select, Contrast notifies you that a mismatch in your selection exists when you select **Add**.  
If you receive this message, you can keep your existing settings, add the appropriate resource to the selected resource groups, or change your action selection.
- Resource groups:** Select one or more resource groups.  
Select a [built-in resource group \(page 1284\)](#) or a [custom group \(page 1287\)](#). If the resource groups you select don't apply to the actions you select **Add**, Contrast notifies you that a mismatch in your selection exists when you select **Add**.  
If you receive this message, you can keep your existing settings, change your action selection, or add the appropriate resource to the selected resource groups.

## Edit custom roles (Preview) ⓘ

Use this procedure to edit a custom role. You cannot change built-in roles.



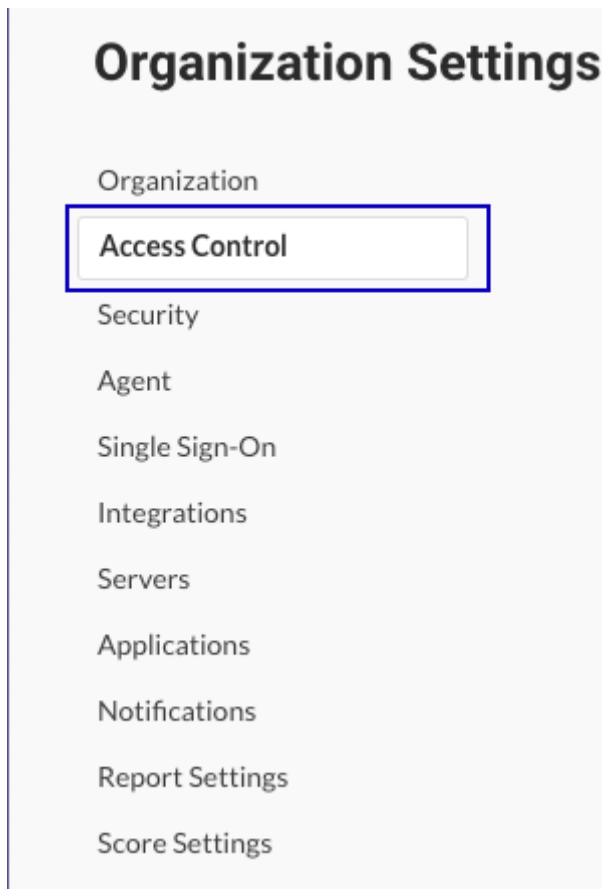
### NOTE

This feature is supported for hosted customers only and is in preview mode. For access to this feature, contact [Contrast support](#).

On-premises customers manage access to Contrast by setting up [organization users and access groups \(page 1163\)](#).

## Steps

1. From the user menu, select **Organization settings**.
2. Select **Access control**.



3. Select the **Roles** tab.
4. To edit an existing role, select the **Edit** icon (✎) at the end of the row for the role you want to change.
5. Change the settings, as needed and select **Save**.

## Copy roles (Preview) ☹

You can copy an existing role, change its settings, and add it as a new role.

Copying a role is useful if you need several roles with the same or similar settings or if you want to use a specific role as a template.

### Before you begin

- **Hosted customers:** You need a role with the Manage platform organization action.
- **On-premises customers:** Manage access to Contrast by setting up [organization users and access groups \(page 1163\)](#).
- You can copy built-in or custom roles.

## Steps

1. From the user menu, select **Organization settings**.
2. Select **Access control**.

3. Select the **Roles** tab.
4. Locate the role you want to copy and select the **Copy** icon (📄) at the end of the row.
5. Change any of the settings and select **Add**.  
Contrast creates a role with the new settings.

## Delete roles (Preview) ☹️

Use this procedure to delete a custom role. You cannot delete built-in roles.



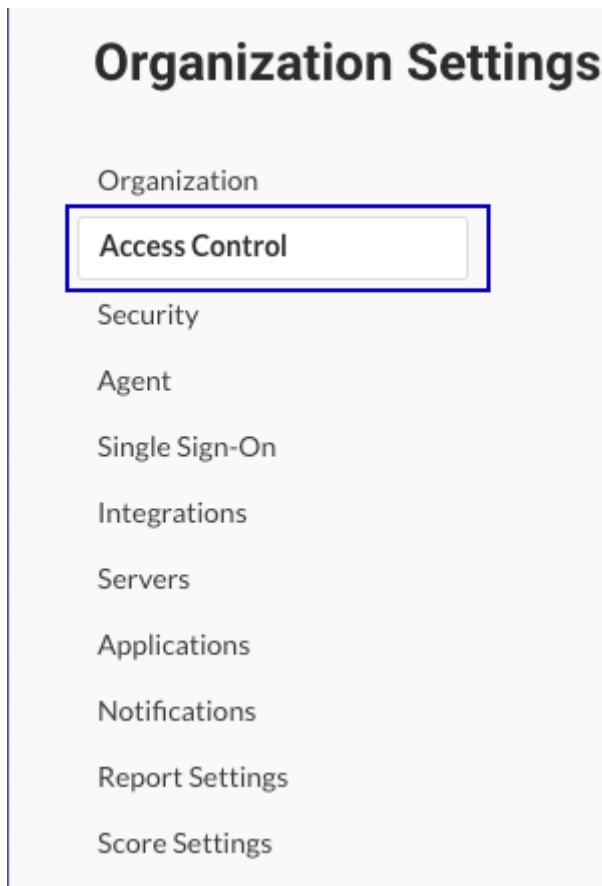
### NOTE

This feature is supported for hosted customers only and is in preview mode. For access to this feature, contact [Contrast support](#).

On-premises customers manage access to Contrast by setting up [organization users and access groups \(page 1163\)](#).

## Steps

1. From the user menu, select **Organization settings**.
2. Select **Access control**.



3. Select the **Roles** tab.
4. Select the **Delete** icon (🗑️) at the end of the row for the role you want to delete.

- In the Delete role window, select **Delete**.

## Users (Preview) ☹

Once you add users to the Users list, you can assign them to specific user access groups.



### NOTE

This feature is supported for hosted customers only and is in preview mode. For access to this feature, contact [Contrast support](#).

On-premises customers manage access to Contrast by setting up [organization users and groups](#). (page 1163)

## Users tab

The Users tab shows details about the Contrast users:

- Name:** The first and last name of the user.
- Access groups:** The user access groups assigned to the user.  
User Access groups determine the roles assigned to the user. The roles determine the actions and resources for the user.
- Type:** The type of account: user, API or guest.  
To copy the credentials for the API user, select the cloud icon (☼) next to the type. Then, select the copy icon (📄) next to the credentials.  
To change a guest account to a standard account, select **Guest** in the Type column and then, select **Add user**.
- Status:** The current status of the user account:
  - Active:** The user has access to Contrast data.
  - Waiting for activation:** Users need to respond to the activation email from Contrast before they can access Contrast data.
  - Inactive:** The account is no longer in use.  
To re-activate the account, select the Activate icon (☹) and select **Activate**.
  - Locked:** The user is locked out of their account based on a security policy.  
To unlock the account, select the check mark next to it and select **Unlock**.
- Last login:** The most recent date and time when the user logged in to Contrast.

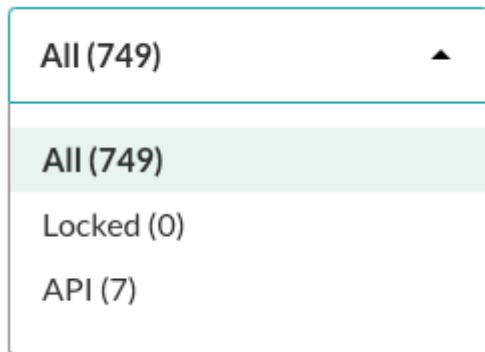
Users   Resource Groups   Roles   User Access Groups						
All (86) ▼		Search		Sort by Name	+ Add User	
NAME (USERNAME)	ACCESS GROUPS	TYPE	STATUS	LAST LOG IN	ACTIONS	
user1@mycompany.com	Organization Administration	—	☑ Active	2023/09/05 02:39 PM	✎ 🗑	
user2@mycompany.com	Organization Administration Developer	API ☼	☑ Active	2023/18/05 11:44 AM	✎ 🗑	
user3@mycompany.com	Organization Administration	—	☑ Active	2023/19/05 04:04 PM	✎ 🗑	
user4@mycompany.com	Organization Administration	—	☑ Active	2023/12/05 09:09 AM	✎ 🗑	
user5@mycompany.com	Organization Administration Developer	API ☼	☑ Active	2023/12/05 01:48 PM	✎ 🗑	



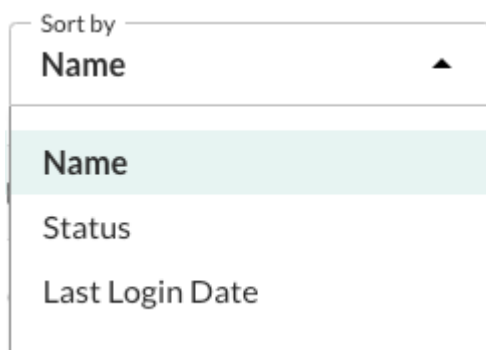
## Filters and sorting

You can refine the view by selecting one of these filters:

- **All:** Displays all users
- **Locked:** Displays users with a locked account.
- **API:** Displays user with API access.



You can sort the view by Name, Status, and Last login date.



## Tasks

From this tab, you can:

- [Add a user \(page 1301\)](#) and assign them to one or more user access groups.
- [Edit a user's details \(page 1303\)](#).
- [Delete a user \(page 1305\)](#).

## Add a user (Preview) ☹️

Use this procedure to add a user.



### NOTE

This feature is supported for hosted customers only and is in preview mode. For access to this feature, contact [Contrast support](#)

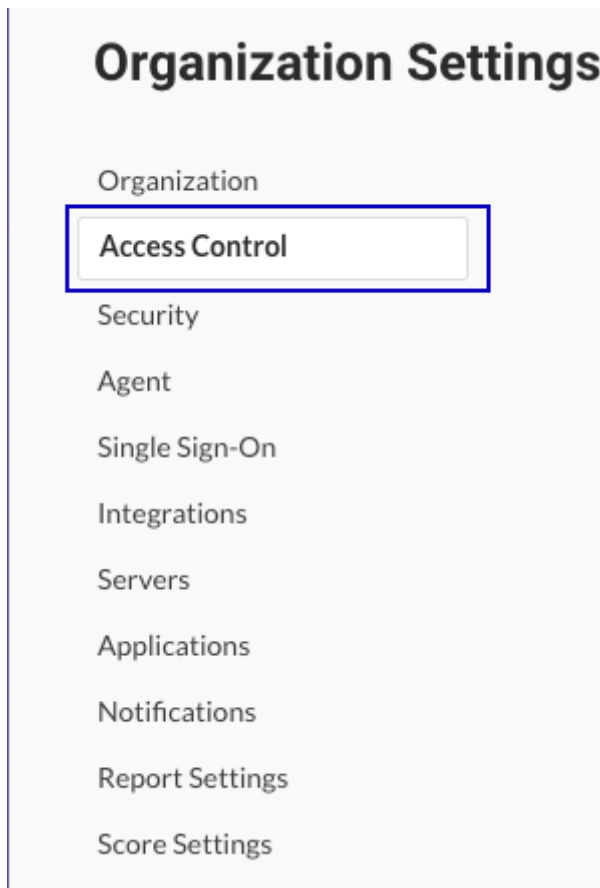
On-premises customers manage Contrast access by setting up [organization users and access groups \(page 1163\)](#).

## Before you begin

- **On-premises users:** Use the [Add or edit a user at an organization level \(page 1163\)](#) procedure.

## Steps

1. From the user menu, select **Organization settings**.
2. Select **Access control**.



3. Select the **Users** tab.
4. Select **Add user**.
5. Specify the user details:

[Users](#) [Resource Groups](#) [Roles](#) [User Access Groups](#)

### Add User

First Name \*

Sally

Last Name \*

Jones

Email \*

sally.jones@mycompany.com

Restrict UI Access

☐ API Only

ALLOW ACCESS

User Access Groups

Organization Edit x

DATE AND TIME PREFERENCES

Date Format

MM/dd/yyyy

Time Format

hh:mm a

Time Zone

(GMT-05:00) Eastern Time (U...

Cancel

Add

User Summary ⓘ

User Access Groups:

Built-in

Organization Edit

- **First name:** The user's first name.
- **Last name:** The user's last name.
- **Email:** The user's email address
- **Restrict UI access:** Select the API only option if you want the user to have access to the Contrast API but not the web interface.  
When you select this setting, a user cannot log in to the Contrast web interface with a password. When you clear the setting, a user can log in with their password and continue to use the API as well.
- **Allow access:** Select a built-in or custom user access group.  
The access group determines the actions and resources available to the user.
- **Date and time preferences:** Select the time format, the date format, and the time zone for the user.

6. Select **Add**.

## Edit a user (Preview) ☹

Use this procedure to edit user details.

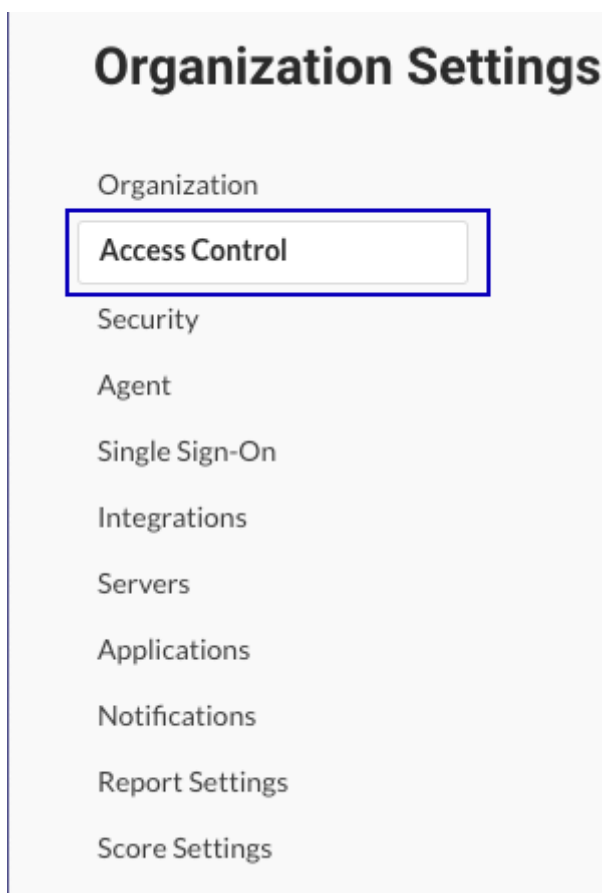
**NOTE**

This feature is supported for hosted customers only and is in preview mode. For access to this feature, contact [Contrast support](#).

On-premises customers manage access to Contrast by setting up [organization users and access groups \(page 1163\)](#).

**Steps**

1. From the user menu, select **Organization settings**.
2. Select **Access control**.



3. Select the **Users** tab.
4. Select the **Edit** icon (✎) at the end of the row for the user whose details you want to change.
5. Change the settings, as needed and select **Save**.  
To change a user's email address, [add a new user \(page 1301\)](#) and include the new address.

**Enable or disable a user account (Preview)**

If a user account is in a **Disabled** state, the user cannot log in to Contrast or use their API keys for authentication. Contrast keeps the original user settings which it applies when you enable the user account again.

**Before you begin**

- You need a role with the Manage organization action.

## Steps

1. From the user menu, select **Organization settings**.
2. Select **Access control**.
3. Select **Users**.
4. Find the user you want to deactivate and select the **Edit** icon (✎) at the end of the row.
5. Under Enable user, clear the **Enabled** checkbox to deactivate a user.  
To enable the user again, select the checkbox.

## Delete a user (Preview) ☹

Use this procedure to delete a user.



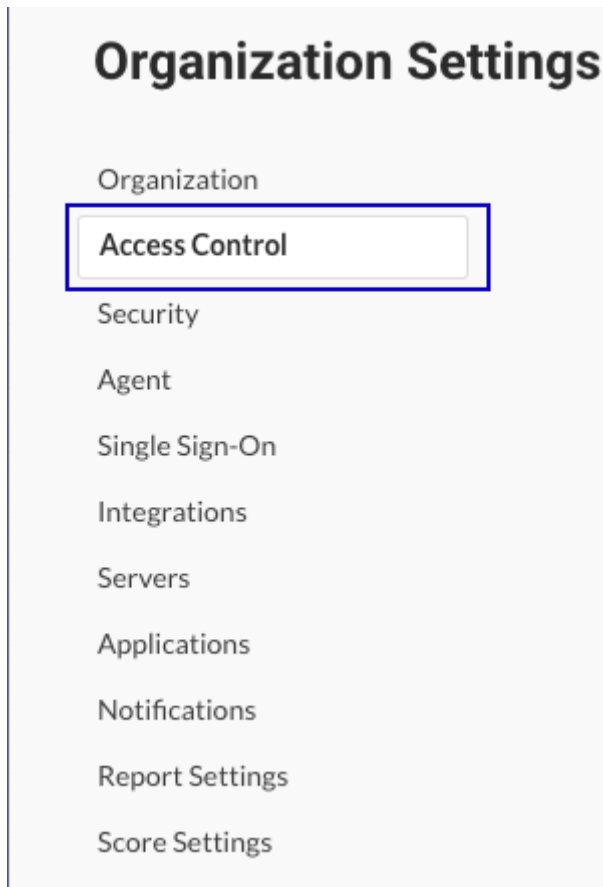
### NOTE

This feature is supported for hosted customers only and is in preview mode. For access to this feature, contact [Contrast support](#).

On-premises customers manage access to Contrast by setting up [organization users and access groups \(page 1163\)](#).

## Steps

1. From the user menu, select **Organization settings**.
2. Select **Access control**.



3. Select the **Users** tab.
4. Select the **Delete** icon (🗑️) at the end of the row for the user you want to delete.
5. In the Delete user window, select **Delete**.

## View my permissions (Preview)

As a Contrast user, you can view your assigned permissions for role-based access control in your user profile. This information is useful if you need to understand which resources you can access and what you can do with those resources.



### NOTE

This feature is supported for hosted customers only and is in preview mode. For access to this feature, contact [Contrast support](#).

On-premises customers manage Contrast access by setting up [organization users and access groups \(page 1163\)](#).

## Steps

1. From the user menu, select **User settings**.
2. Select **My permissions**.
3. To sort the Resource or Actions columns, select the **Sort** icon (↑).

## User permission details

Contrast displays these details for the resources and actions assigned to you:

- **Resource:** A resource is an application or Scan project that you can access.
- **Actions:** An action represents what you can do with the assigned resource.  
[Actions and permissions \(page 1278\)](#) provides additional details about what each action lets you do with a resource.

## View user permissions (Preview)

As an administrator, you can view the permissions and settings for users in your organization.



### NOTE

This feature is supported for hosted customers only and is in preview mode. For access to this feature, contact [Contrast support](#).

On-premises customers manage Contrast access by setting up [organization users and access groups \(page 1163\)](#).

## Before you begin

- A role with any of these actions: Edit organization, Manage organization rules, or Manage organization.

## Steps

1. From the user menu, select **Organization settings**.
2. Select **Access control**.
3. Select **Users**.
4. For a specific user, select the **Key** icon (🔑).
5. To sort the view, select the **Sort** icon (↑) next to any of the column headings.
6. To view details about the user's resource groups, roles, or user access groups, select the link in the column.

You can change the details for any custom resource group, role, or user access group.

## User permission details

For each user, Contrast displays these details:

- **Resource:** The applications and Scan projects the user can access.
- **Resource group:** The resource group that includes the resources a user can access.
- **Actions:** The actions the user can perform for each resource.  
[Actions and permissions \(page 1278\)](#) provides additional details about what each action lets users do with a resource.
- **Roles:** The role assigned to the user that includes their assigned actions and resource groups.
- **User access group:** The user access group associated with the user's role.

## Create an API only user (Preview) ☹

Create an API user account that you can use for all plugins or integrations.

**NOTE**

This procedure is for hosted customers who have role-based access turned on.

If you are an on-premises customer or do not have role-based access turned on, use this [Create API user \(page 1165\)](#) procedure.

**Best practice:** Add a user account that's only purpose is for use with plugins and integrations. Doing so avoids a situation where a user leaves and you delete that user's account. The deletion of that account would result in breaking the plugins and integrations that you use.

An API only account does not receive email notifications, even if the notification settings are turned on.

**Before you begin**

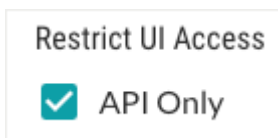
- API users can access Contrast's REST API but cannot log in to the Contrast web interface.
- If you configured your organization to use SAML-based single sign-on (SSO), you can still create an API user.
- Access control guidelines:
  - If you have role-based access control turned on and you need an API only user to run scripts that pull data, the View application action should be sufficient.
  - If you have role-based access control turned on and you need an API only user to remediate vulnerabilities, add applications to Contrast, or run scans, the Edit application action should be sufficient.
  - If you are using role-based access control, assign the user access groups that include the relevant applications and projects to the API only user.
  - Avoid assigning Administrator actions to an API only user. Administrator actions and roles provide additional permissions that an API only user doesn't usually need.

**Steps**

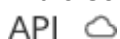
1. From the user menu, select **Organization settings**.
2. Select **Access control**.
3. Select the **Users** tab.
4. Select **Add user**.




5. Enter the user's first name, last name, and email address.
6. Select the **API access** option.



7. Select **Add**.
8. In the Users list, verify you see the new user with the API label in the Type column.



9. To use the API access account, get the connection strings:
  - a. In the Users list, select the API icon  in the Type column and copy the Service key and Authorization Header.

The Service key is unique to each individual user. The value of the Authorization Header contains `base64(username:service_key)`.



Other keys, such as the Organization ID and API key are shared across all users.

- b. Use these credentials when you use a Contrast API.

## User access groups (Preview)

User access groups let you specify specific roles and resource groups for one or more users. Roles and resource groups define the permissions for users.

You can add users to built-in user access groups or create [custom groups \(page 1310\)](#).



### NOTE

This feature is supported for hosted customers only and is in preview mode. For access to this feature, contact [Contrast support](#).

On-premises customers manage access to Contrast by setting up [organization users and access groups \(page 1163\)](#).

## User access groups tab

The User access groups tabs displays the list of existing groups. From this tab, you can:

- View a list of user access groups.  
Use search to find specific groups.
- [Add a user access group \(page 1310\)](#)
- [Edit a user access group \(page 1312\)](#)
- [Delete a user access group \(page 1313\)](#)

**Organization Settings**

- Organization
- Groups
- Users
- Access Control**
- Security
- Agent
- Single Sign-On
- Integrations
- Servers
- Applications
- Notifications
- Report Settings
- Score Settings

Users   Resource Groups   Roles   **User Access Groups**

**User Access Groups**

[+ Add Group](#)

GROUP		DESCRIPTION	MEMBERS	
DevOps Administrators	Built-in	Users who manage application settings and projects.	0	
AppSecurity Administrators	Built-in	Users who manage applications and projects, as well as view organization data.	0	
Organization viewers	Built-in	Users who view organization data.	0	
Organization editors	Built-in	Users who view and edit organization data, such as view agent keys, manage notifications, and manage score settings.	0	
Ecommerce		Ecommerce team	1	

## Built-in user access groups

You can add users to these built-in user access groups:

This built-in group:	Includes these built-in roles:
Organization viewers	Organization viewer

This built-in group:	Includes these built-in roles:
Organization editors	Organization editor
Organization rules administrators	Organization rules administrator
Organization administrators	Application administrator
	Project administrator
	Organization administrator
	Protect viewer
	Serverless user
AppSecurity administrators	SCA project group administrator
	App Security administrator
	DevOps administrator
	Organization viewer
	Protect viewer
DevOps administrators	Serverless user
	Scan uploader
	Application viewer
	Project viewer
	SCA project group viewer
Developers	Project viewer
Scan manager	Protect viewer
Protect viewers	Serverless user
Serverless users	

**NOTE**

Other than adding or removing users, you cannot change the settings for the built-in user access groups.

**See also**

[Built-in roles \(page 1291\)](#)

**Add user access groups (Preview) ☹**

Create user access groups to let a collection of users access projects, applications, and organization settings. The roles you assign to users in the group determine which capabilities they can access.

**NOTE**

This feature is supported for hosted customers only and is in preview mode. For access to this feature, contact [Contrast support](#).

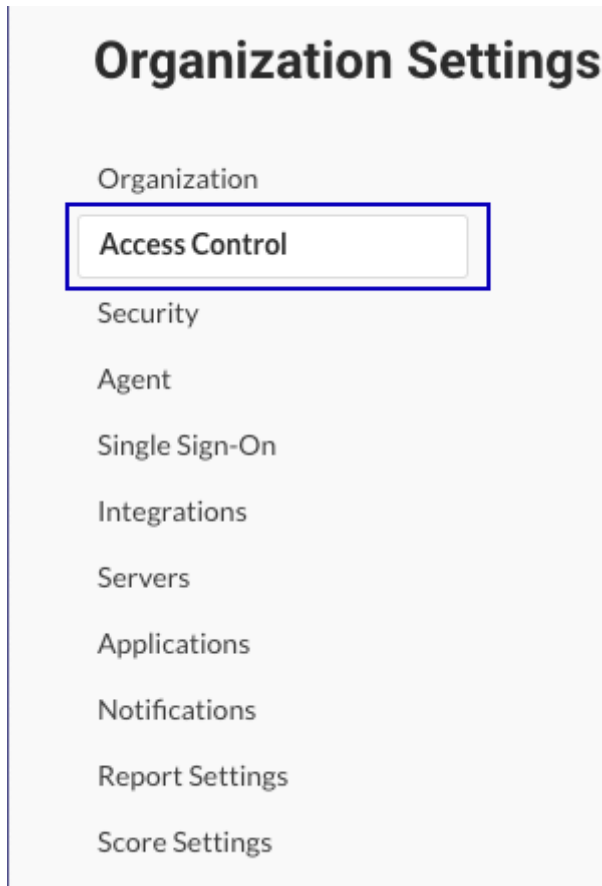
On-premises customers manage access to Contrast by setting up [organization users and access groups \(page 1163\)](#).

## Before you begin

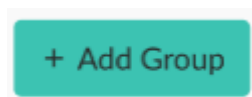
- [Add users \(page 1301\)](#), if necessary.
- Determine whether you need to [add custom roles \(page 1295\)](#) or [custom resource groups \(page 1287\)](#).

## Steps

1. From the user menu, select **Organization settings**.
2. Select **Access control**.



3. In the User access groups tab, select **Add group**.



4. In the Add user access group window, specify these settings:

**Organization Settings**

Organization

Access Control

Security

Agent

Single Sign-On

Integrations

Servers

Applications

Notifications

Report Settings

Score Settings

Users Resource Groups Roles **User Access Groups**

### Add user access group

User access groups assign users to roles. For easier management of these groups, use descriptive names and easy-to-identify descriptions.

User access group name \*  
Ecommerce

User access group description \*  
Ecommerce team

Users \*  
user1@mycompany.com x user2@mycompany.com x

Roles \*  
Developer x

**Your user access group summary** ⓘ

User access group name  
Ecommerce

User access group description  
Ecommerce team

Users  
user1@mycompany.com  
user2@mycompany.com

Roles  
Developer

Cancel Add

- **User access group name:** Specify a name for the user group.  
The name must be unique for the organization. Use up to 255 characters, including spaces and special characters.
- **User access group description:** Specify a description of the group.  
Consider using a description that clearly identifies the purpose of the group. Use up to 1024 characters, including spaces and special characters.
- **Users:** Specify users for the group.  
To display a list of users, select the triangle ▾ at the end of the box or start typing a name.  
If you don't see specific users, go to the **Users** page under **user menu > Organization settings** and verify that they are Contrast organization users.
- **Roles:** From the dropdown, select one or more roles to apply to all users in the group.  
You can select [built-in roles \(page 1291\)](#) or [custom roles \(page 1295\)](#).

5. Select **Add**.

## Edit user access groups (Preview) ⓘ

Use this procedure to edit a user access group.

For built-in user access groups, you can only change the users assigned to the group.



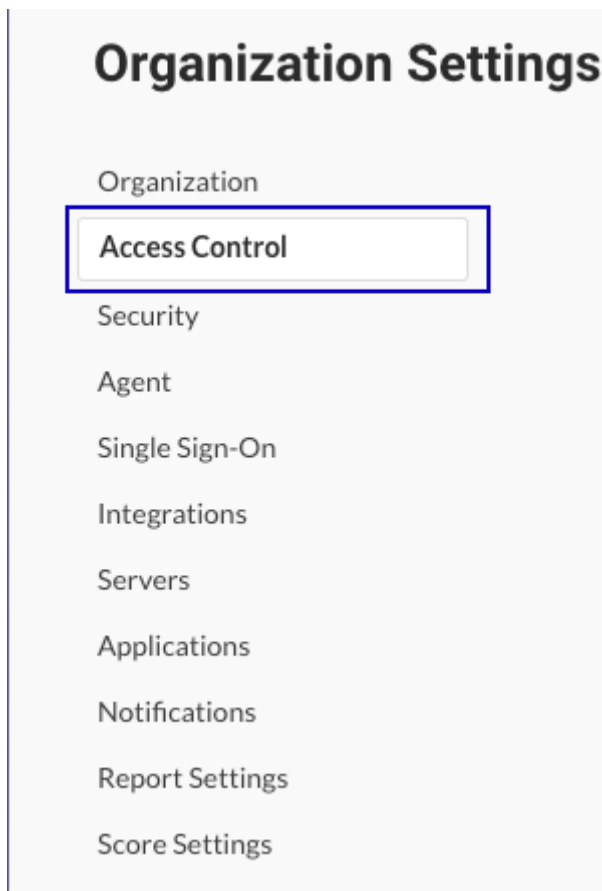
### NOTE

This feature is supported for hosted customers only and is in preview mode. For access to this feature, contact [Contrast support](#).

On-premises customers manage access to Contrast by setting up [organization users and access groups \(page 1163\)](#).

## Steps

1. From the user menu, select **Organization settings**.
2. Select **Access control**.



3. Select the **User access groups** tab.
4. Select the **Edit** icon (✎) at the end of the row for the group you want to change.
5. Change the settings, as needed and select **Save**.

## Delete custom user access groups (Preview) ☹️

Use this procedure to delete a custom user access group. You cannot delete built-in user access groups.



### NOTE

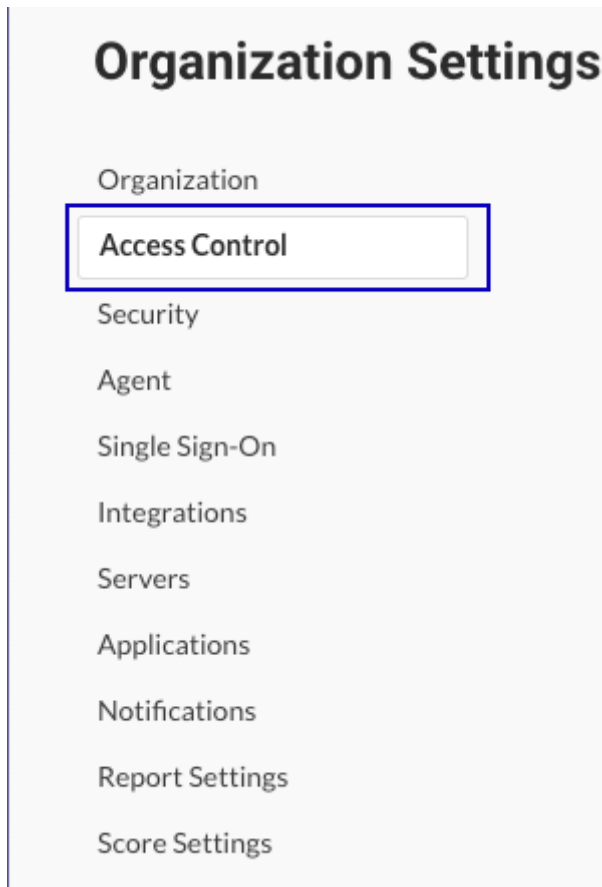
This feature is supported for hosted customers only and is in preview mode. For access to this feature, contact Contrast support.

On-premises customers manage access to Contrast by setting up [organization users and access groups \(page 1163\)](#).

## Steps

1. From the user menu, select **Organization settings**.

2. Select **Access control**.



3. Select the **User access groups** tab.
4. Select the **Delete** icon (🗑️) at the end of the row for the group you want to delete.
5. In the Delete user access group window, select **Delete**.

## Authorization troubleshooting for access control (Preview) 🕒



### NOTE

Role-based access control is supported for hosted customers only and is in preview mode. If you want to be an early adopter, contact [Contrast support](#).

## Authorization for access control

Authorization for access control determines the actions a user can take for a specific resource. If a user receives a 403 (Forbidden message), Contrast access control has determined that the user does not have enough permissions to perform the requested operation. Access control considers these elements during authorization:

- **User:** The person performing a request.
- **Action:** The operation you want to perform. For example, viewing, editing, or managing.
- **Resource:** The resource that access control protects from unauthorized access. For example, applications, Scan projects, user access groups, resource groups, and roles.

## Access control: troubleshooting steps

1. Check the user permissions.
  - a. Under the user menu, select Organization settings.
  - b. Select **Access control**.
  - c. In the Users tab, find the user in the list and select the key icon (🔑) in the Actions column.
  - d. If the user is assigned the correct roles and resources, go to the next step. Otherwise, update the user settings.

You can also retrieve a user's role-based access control permissions by using the [Contrast API](#).

2. (Optional) Verify effective user permissions.

The Contrast web interface lets you view permissions and settings for individual users, however, you can also use the [Contrast API](#) to verify the effective user permissions, such as:

- Resources (applications, Scan projects, and functions) that the user can access.
- The actions assigned to the user.
- The role, user access group, and resource group combination that provides the user permissions.

[Access control queries \(page 1315\)](#) provides examples of how to use the Contrast API to check user settings and permissions.

If the user settings and permissions look correct, contact [Contrast support](#) for additional help. Otherwise, update the user settings and permissions.

## Examples: Access control queries for troubleshooting (Preview) 🔄



### NOTE

This feature is supported for hosted customers only and is in preview mode. If you want to be an early adopter, contact Contrast support.

This topic provides examples of how to use the [Contrast API](#) to get access control details when you troubleshoot authorization issues.

To use the API to get role-based access control information for a user, you need a role that includes the Manage organization and Manage platform organization actions.

### User ID query

Use this query to get a user's ID.

1. Set these environment variables:

```
HOSTNAME=http://<YourHostName>
ADMIN_APIKEY=<ServiceKey>
ADMIN_AUTH=<AdminAuthorization>
ORGID=<OrganizationId>
USER_EMAIL=<UserEmail>
USERID=<obtained in the next step>
```

In the Contrast web interface, you can get variable values under the **user menu > User settings** for the administrator or the user.

- Replace <YourHostName> with the URL for your Contrast instance. For example: `https://mycompany.com/Contrast`.

- Replace <ServiceKey> with the Service key for the administrator.
- Replace <AdminAuthorization> with the Authorization header for the administrator.
- Replace <OrganizationID> with the ID for the organization where the user is a member.
- Replace <UserEmail> with the user's email address used to log in to Contrast.

2. To find the user's ID, run this query.

```
curl -X GET --location "$HOSTNAME/api/v4/organizations/$ORGID/users/$USER_EMAIL" \
 -H "API-Key: $ADMIN_APIKEY" \
 -H "Authorization: $ADMIN_AUTH"
```

**Sample response with the user ID highlighted:**

```
{
 "userId": "4790deb8-972d-47c8-b2d0-219617999c83",
 "username": "contrast_view",
 "organizationId": "2f95790d-64dd-4344-9b1c-920021d112bb",
 "firstName": "NX374ERI11",
 "lastName": "KFG0S17TX6",
 "status": "ACTIVE",
 "type": "STANDARD",
 "enabled": true,
 "language": "en",
 "dateTimePreferences": {
 "dateFormat": "MM/dd/yyyy",
 "timeFormat": "hh:mm a",
 "timeZone": "EST"
 },
 "auditDates": {
 "lastLoginTime": "2024-05-29T14:28:00.000+00:00",
 "creationDate": "2024-04-23T20:33:21.000+00:00"
 },
 "userAccessGroupMembership": [
 {
 "userAccessGroupId": "725e7af7-8cf4-44b3-a6d2-b30e6df6573e"
 },
 {
 "userAccessGroupId": "eeb65497-1e65-4eb3-99af-b781a4ce7d29"
 },
 {
 "userAccessGroupId": "12565428-1063-41c4-ada0-cbee082f5eca"
 }
],
 "apiOnly": false,
 "external": false,
 "serviceKey": "demo"
}
```

3. Assign the user ID to the USERID variable:

```
USERID=4790deb8-972d-47c8-b2d0-219617999c83
```

### User access query

Run this query to get all access control details for a user.

```
curl -X GET --location "$HOSTNAME/api/v4/organizations/$ORGID/access-control-query/users/$USERID" \
```



```
-H "API-Key: $ADMIN_APIKEY" \
-H "Authorization: $ADMIN_AUTH"
```

### Login permission query

Run this query to determine if a user has login permissions.

The user must have a role that includes at least the View organization action. Run this query:

```
curl -s -X GET --location "$HOSTNAME/api/v4/organizations/$ORGID/access-
control-query/users/$USERID" -H "API-Key: $ADMIN_APIKEY" -H "Authorization:
$ADMIN_AUTH" \
 | jq | grep ORG_SETTINGS | sort | uniq | wc -l | if grep -q 0; then
echo "Not configured for login"; else echo "Configured for login"; fi
```

#### Sample result:

```
Configured for login
```

### User role query

Run this query to find all the roles assigned to a user.

```
curl -s -X GET --location "$HOSTNAME/api/v4/organizations/$ORGID/access-
control-query/users/$USERID" -H "API-Key: $ADMIN_APIKEY" -H "Authorization:
$ADMIN_AUTH" \
 | jq | grep roleName | sort | uniq
```

#### Sample result:

```
"roleName": "ORGANIZATION_VIEW_ROLE",
"roleName": "rg with 1 app role",
```

### User access group query

Run this query to find all the user access groups assigned to the user.

```
curl -s -X GET --location "$HOSTNAME/api/v4/organizations/$ORGID/access-
control-query/users/$USERID" -H "API-Key: $ADMIN_APIKEY" -H "Authorization:
$ADMIN_AUTH" \
 | jq | grep userAccessGroupName | sort | uniq
```

#### Sample result:

```
"userAccessGroupName": "Organization View",
"userAccessGroupName": "rg with 1 app uag",
```

### Resource types, actions, and ID query

Run this query to get a list of of resources, actions and IDs for a user.

```
curl -s -X GET --location "$HOSTNAME/api/v4/organizations/$ORGID/access-
control-query/users/$USERID" \
 -H "API-Key: $ADMIN_APIKEY" \
 -H "Authorization: $ADMIN_AUTH" | jq '.accessList[] | "\(.resourceType)
\(.actions) \(.resourceId)"' | sort | uniq
```

#### Sample result:

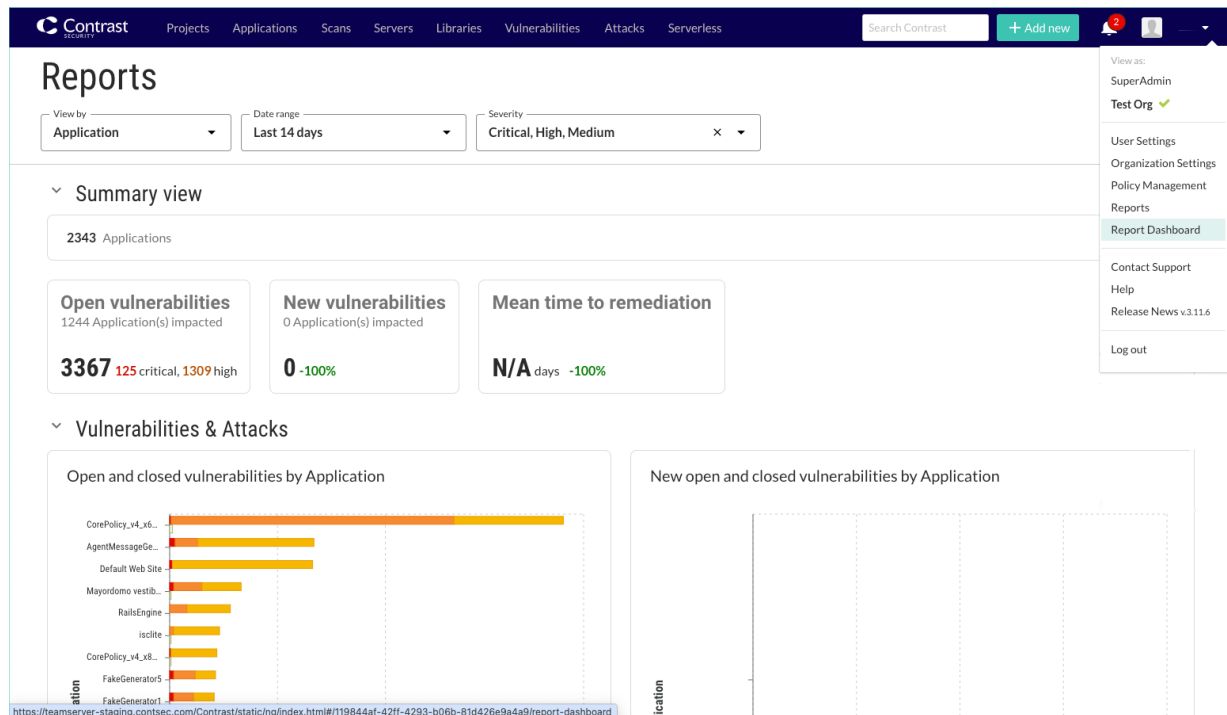
```
"APPLICATION
[\"APPLICATION_EDIT\", \"APPLICATION_RULES_ADMIN\", \"APPLICATION_ADMIN\", \"AP
PLICATION_VIEW\"] 0383916c-956b-418b-a114-0ffc3420cb1c"
```

```
"APPLICATION
[\"APPLICATION_EDIT\", \"APPLICATION_RULES_ADMIN\", \"APPLICATION_ADMIN\", \"APPLICATION_VIEW\"] 1a7539c4-2339-4524-958d-b60482ebf1f8"
"APPLICATION
[\"APPLICATION_EDIT\", \"APPLICATION_RULES_ADMIN\", \"APPLICATION_ADMIN\", \"APPLICATION_VIEW\"] 2762d3b4-1033-406e-94cc-f4040b6e7111"
"APPLICATION
[\"APPLICATION_EDIT\", \"APPLICATION_RULES_ADMIN\", \"APPLICATION_ADMIN\", \"APPLICATION_VIEW\"] 7843ab03-84a7-4f21-93bf-37e7bf47f94d"
"APPLICATION
[\"APPLICATION_EDIT\", \"APPLICATION_RULES_ADMIN\", \"APPLICATION_ADMIN\", \"APPLICATION_VIEW\"] 97b1a6a1-ee34-4974-9ee2-4c92733de2bb"
"APPLICATION
[\"APPLICATION_EDIT\", \"APPLICATION_RULES_ADMIN\", \"APPLICATION_ADMIN\", \"APPLICATION_VIEW\"] a8b804ad-cbe4-43e7-a03f-3f639f5680ab"
"APPLICATION
[\"APPLICATION_EDIT\", \"APPLICATION_RULES_ADMIN\", \"APPLICATION_ADMIN\", \"APPLICATION_VIEW\"] d0c389d7-744c-4f2a-b391-8bf41c0dc09c"
"APPLICATION
[\"APPLICATION_EDIT\", \"APPLICATION_RULES_ADMIN\", \"APPLICATION_ADMIN\", \"APPLICATION_VIEW\"] de9248b1-e8f5-43b9-a4e3-8a65e844691d"
"APPLICATION [\"PROTECT_ACCESS\", \"PROTECT_POLICIES_MANAGE\"]
0383916c-956b-418b-a114-0ffc3420cb1c"
"APPLICATION [\"PROTECT_ACCESS\", \"PROTECT_POLICIES_MANAGE\"]
1a7539c4-2339-4524-958d-b60482ebf1f8"
"APPLICATION [\"PROTECT_ACCESS\", \"PROTECT_POLICIES_MANAGE\"]
2762d3b4-1033-406e-94cc-f4040b6e7111"
"APPLICATION [\"PROTECT_ACCESS\", \"PROTECT_POLICIES_MANAGE\"]
7843ab03-84a7-4f21-93bf-37e7bf47f94d"
"APPLICATION [\"PROTECT_ACCESS\", \"PROTECT_POLICIES_MANAGE\"] 97b1a6a1-ee34-4974-9ee2-4c92733de2bb"
"APPLICATION [\"PROTECT_ACCESS\", \"PROTECT_POLICIES_MANAGE\"] a8b804ad-cbe4-43e7-a03f-3f639f5680ab"
"APPLICATION [\"PROTECT_ACCESS\", \"PROTECT_POLICIES_MANAGE\"]
d0c389d7-744c-4f2a-b391-8bf41c0dc09c"
```

## Report dashboard (Preview)

The report dashboard lets you view aggregated statistics for vulnerabilities in your organization. It also provides details that help you understand your organization's progress with managing and fixing vulnerabilities.

If you are using [role-based access control \(page 1277\)](#), the report shows details based on the resources associated with your role.



## Report dashboard details

### • Summary view

- If you select Application in the View by filter, the summary shows the total number of applications included in this report. If you select [application metadata \(page 1174\)](#), the summary also shows the number of values found for the selected metadata. .
- **Open vulnerabilities:** The number of vulnerabilities that have no closed date or have a closed date that occurred after the selected time frame.
- **New vulnerabilities:** The number of vulnerabilities that Contrast first detected during the selected time frame.
- **Mean time to remediation:** The average time it took to close a security vulnerability during the selected time frame.

### • Vulnerabilities and attacks

To see additional details, hover over the bars in each chart.

- **Open and closed vulnerabilities:** All open and closed vulnerabilities as of the current date.
- **New open and closed vulnerabilities:** Vulnerabilities that were opened or closed during the selected time frame. vulnerabilities
- **Top vulnerability types to fix:** The vulnerabilities that Contrast considers the most important ones to fix.
- **Open and closed rates:** Trends that show the rate at which teams close open vulnerabilities

### • Application or metadata group

This section shows details for all applications or the applications that use the selected metadata. To view additional details, hover over the bars in the chart.

- **Name or Value:** The application name or the value for selected metadata.
- **Onboarded apps:** The number of applications added to Contrast or the number of applications in Contrast that use the selected metadata.
- **Open vulns:** The number of vulnerabilities with an open status in the specified time frame
- **New vulns:** The number of vulnerabilities that Contrast reported in the specified time frame
- **Closed:** The number of vulnerabilities that closed in the specified time frame.
- **Mean time to remediation:** The average time it took to fix the vulnerabilities in the specified time frame.

## See also

[View report dashboard \(page 1320\)](#)

## View report dashboard (Preview)

### Before you begin

- **User and groups:** If you are using [users and groups \(page 1163\)](#) for access control, you need an Organization Admin role.
- **Role-based access control:** If your organization has [role-based access control \(page 1277\)](#) turned on, you need a role with the View application action. The dashboard shows details based on the resource groups associated with your role.

### Steps

1. From the user menu, select **Report dashboard**.
2. Use these filters to refine the view:
  - **View by:** Select **Application** to create a view of all application vulnerability details or select [application metadata \(page 1174\)](#) to view details based on the metadata.
  - **Date range:** Select a time frame or select **Custom** to create a customized time frame.
  - **Severity:** Select one or more vulnerability severity levels.

## Export Aggregated Vulnerability report (Preview)

You can export the details on the report dashboard to a PDF file.

### Before you begin

- **User and groups:** If you are using [users and groups \(page 1163\)](#) for access control, you need an Organization Admin role.  
The report includes all displayed details.
- **Role-based access control:** If your organization has [role-based access control \(page 1277\)](#) turned on, you need a role with the View application action.  
The report includes details based on the resources associated with your role.  
Role-based access control is a preview feature and not available to all users. If you want to use it, contact your Contrast representative.

### Steps

1. From the user menu, select **Report dashboard**.
2. Optionally, select filters to refine the view.
3. Select **Export PDF** at the top of the page.
4. Specify a download location when prompted to do so.

## Contrast Visual Studio Code plugin (Preview)



### NOTE

Integration is to be used with [Contrast Scan \(page 33\)](#).

Use the [Contrast Visual Studio Code](#) IDE (Integrated Development Environment) plugin to integrate security vulnerabilities with projects ([Scan \(page 33\)](#)). The Contrast plugin lets developers view the vulnerabilities identified during the most recent security scans and take remedial actions. The Contrast plugin provides thorough information about vulnerabilities associated with projects on a near real-time basis. The plugin offers filters based on severity, status, and discovery date to customize the vulnerability data to view.

The key features include a vulnerability report to view vulnerabilities associated with applications for projects in Scan, provide a tree view on the list of the vulnerabilities related to the current file open in the IDE with visual indicators based on criticality, see which line of code has a vulnerability through visual indicators, automate the collection of vulnerabilities for applications and projects based on a schedule.

## Before you begin

- Make sure you have the supported system requirements:
  - CPU: Quad-core
  - RAM: 16 GB
  - Storage: SSD, 128 GB
  - Monitor: 1080p
- Make sure you have the supported software requirements:
  - Operating systems: Ubuntu 22.04.5 LTS or Windows 11
  - Node: 20.17.0
  - VSCode: 1.93.0 and above

## Download Visual Studio Code

1. Go to [Visual Studio Code](#).
2. Click the **Download** button. The website will automatically detect your operating system and provide the appropriate installer.
  - Windows: `.exe` file
  - Mac: `.dmg` file
  - Linux: `.deb`, `.rpm`, or `.tar.gz` file
3. Continue by installing the Visual Studio Code.

## Install Visual Studio Code

### Linux

1. Open a terminal and go to the directory where you downloaded the file.
2. Run the command:

```
bash: sudo apt install ./code_*.deb
```

3. Continue by launching Visual Studio code.

### Windows

1. Go to the directory where you downloaded the file and open the `.exe` file.
2. Follow the installation wizard:
  - Accept the license agreement
  - Choose the installation folder
  - Select optional tasks like adding VS Code to the PATH
3. Click **Install** and then **Finish**.

4. Continue by launching Visual Studio code.

## macOS

1. Go to the directory where you downloaded the file and open the `.dmg` file.
2. Drag the *Visual Studio Code* icon into the **Applications** folder.
3. Continue by launching Visual Studio code.

## Launch Visual Studio Code

1. Locate and open Visual Studio Code.
  - **Linux:** Use your desktop menu or run code from the terminal.
  - **Windows:** Search for **Visual Studio Code** under the **Start** menu.
  - **Mac:** Open it from the **Applications** folder.
2. Continue by installing the Contrast plugin.

## Install the Contrast Visual Studio Code plugin

Select one of these options.

### Install the plugin via Visual Studio Marketplace

Follow the steps from the [marketplace](#).

1. Search for and then open **VS Code**.
2. Go to the **Extensions** view.
3. Search for **Contrast IDE** and select it to view more details.
4. Click the **Install** button on the extension's page.
5. If required, restart the VSCode after installing the Contrast extension.

### Install the plugin via manual installation

1. Launch Visual Studio Code on your system.
2. Go to the **Extensions** view.
3. Click the *more options* icon ( `:` ) in the top-right corner and select **Install from VSIX**.
4. Go to the location of the `.vsix` file and select it to install the plugin.