Contrast SECURITY

Contrast Documentation

January 18, 2023

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

Copyright © 2023 Contrast Security

Table of Contents

- Get started 10
  - Customization 10
  - Next steps 12
  - How Contrast works 12
    - How Contrast integrates with your development environment 13
    - Analysis techniques and data sources 13
    - Contrast agents 13
    - Agent configuration 13
    - Static scans 14
    - Protection for cloud-native applications 14
    - Integrations 14
  - Contrast walk through 14
    - Customize the Contrast environment 15
    - Step 1: Configure applications for security testing 17
    - Step 2: Configure applications to block attacks 19
    - Step 3: Fix code and retest applications 21
  - Assess 21
    - Features 21
    - Customization 21
  - SCA 22
    - Features 22
    - Contrast data 23
  - Protect 23
    - How Protect works 23
    - Customization 23
  - Scan 24
    - Features 24
    - See also 24
  - Serverless Application Security 24
    - Features 25
    - Benefits 25
    - How it works 25
    - Security and privacy 26
    - See also 26
  - Performance 27
  - Community Edition (CE) 27
    - Community Edition features 27
    - Community Edition portal 27
    - Next steps 28
- Agents 29
  - Install and configure an agent 30
    - Installation 30
    - Configuration 30
    - Order of precedence 31
    - Find the agent keys 32
    - YAML configuration 33
    - Agent configuration editor 34
    - Agent Operator (Kubernetes operator) 36
    - Environment variables 51
  - Java (Kotlin, Scala) 52
    - Supported technologies 52
    - System requirements 54
    - Install 54

Configure 70
Legacy Java agent 125
Java agent telemetry 125
.NET Framework 126
Supported technologies 126
System requirements 127
Install 128
Configure 140
Legacy .NET Framework agent 170
Use with IIS Express 172
Use with Azure 173
Azure Service Fabric 173
Profiler chaining 174
.NET Framework Contrast tray 176
Application pools 178
Telemetry 179
.NET Core 180
Supported technologies 181
System requirements 182
Install 182
Configure 197
Profiler chaining 224
Telemetry 226
Azure functions 227
Node.js 229
Contrast service 229
Supported technologies 230
System requirements 231
Install 232
Configure 239
Rewriter CLI 259
Use the Node.js agent with ESM 261
Transpilers, compilers and source maps 261
Telemetry 262
PHP 263
Supported technologies 263
System requirements 263
Install 263
Configure 266
Python 294
System requirements 295
Supported technologies 295
Install 296
Configure 298
Telemetry 352
Ruby 353
System requirements 353
Supported technologies 354
Install 355
Configure 357
Telemetry 409
Go 409
Supported technologies 410
System requirements 410
Install 410

Configure 415
Contrast service 430
Install 430
Configure 431
Install 432
Configure 432
Use Contrast 434
User settings 435
Log in 435
Change password 435
Two-step verification 436
Manage profile 436
View API keys 436
Manage notifications 437
View permissions 437
Applications 438
View 438
Edit application settings 441
Tag 441
Merge 442
Archive 443
Reset 444
Delete 444
Session metadata 445
Route coverage 448
Flow maps 453
Scans 455
Scan process 455
Contrast Scan supported languages 457
Scan package preparation 458
Create a scan project 459
Start a scan 460
Cancel a scan 461
Monitor scans 461
Contrast Scan local engine 462
Analyze results 468
View scan policies 479
Change scan settings 479
Archive scan projects 479
Unarchive scan projects 480
Integrate scans with build pipelines 480
Servers 483
Server settings 483
Settings in a configuration file 484
Agent configuration instructions 484
Contrast options 484
Configure 484
Output to syslog 486
Libraries 488
See also 488
View libraries 489
Discover or delete 490
Tags 491
Track 492
Runtime library usage 492

Export ... 494
Open-source licenses ... 495
View dependency trees ... 495
Library scoring guide ... 496
CVE search ... 497
Serverless ... 497
See also ... 498
Inventory ... 498
Supported languages for Contrast Serverless ... 498
Supported platform for Contrast Serverless ... 498
Scan types and monitoring ... 498
Get started with Contrast Serverless ... 499
Scan functions on demand ... 500
View results ... 501
Change inventory criteria ... 506
Change serverless scan settings ... 507
View function and service relationships ... 507
Block accounts ... 510
Uninstall ... 510
Contrast CLI ... 511
Before you start ... 511
About Contrast CLI ... 511
Install Contrast CLI ... 511
Run Contrast CLI ... 512
Contrast CLI commands ... 513
Legacy Contrast CLI ... 518
Vulnerabilities ... 524
View for organization ... 525
View application vulnerabilities ... 526
View vulnerability rates ... 528
Add and delete vulnerabilities ... 529
Group vulnerabilities by sink ... 529
Merge vulnerabilities ... 530
Tag ... 530
Track ... 531
Events ... 533
Fix ... 533
Export ... 534
Status ... 535
Edit severity ... 539
Attacks ... 539
Event data retention ... 539
Tasks ... 540
View attacks ... 540
Monitor attacks ... 542
Manage attacks ... 543
Add tags to attacks ... 545
Run attack scripts ... 546
Reports ... 547
Attestation reports ... 547
Compliance reports ... 548
DISA STIG Viewer checklists ... 549
Software bill of materials ... 549
Vulnerability trend reports ... 550
Organization statistics ... 552

Integrations ... 553
Cloud integrations ... 553
Chat tools ... 554
Code repository integrations ... 554
Continuous integration and build tools ... 556
Enterprise and extensibility integrations ... 557
IDE plugins ... 559
Incident management systems ... 560
SIEM tools ... 561
Work tracking platforms ... 562
Azure Boards ... 563
See also ... 563
Connect ... 564
Auto create tickets ... 564
Two-way integration ... 565
Personal access tokens ... 565
Agile Central ... 566
Manage credentials ... 567
Azure Pipelines ... 567
Install and configure ... 567
Configure a task ... 568
Add a release gate ... 568
Azure Service Fabric ... 569
Bamboo ... 571
Install ... 571
Configure thresholds ... 571
Bugzilla ... 572
Eclipse ... 572
Generic webhooks ... 573
Generic webhooks ... 573
Variables ... 575
Events and variables ... 578
GitHub ... 578
Gradle ... 579
Sample application ... 579
Use the plugin ... 580
Jenkins ... 580
Install and use Jenkins plugin ... 581
See also ... 581
Define a connection ... 581
Jenkins security controls ... 582
Define a job outcome policy ... 585
Run a build ... 589
Jira ... 589
See also ... 589
Connect to Jira ... 589
Configure Jira for Assess ... 590
Configure Jira for Serverless ... 591
Manage credentials ... 592
Maven ... 593
Goals ... 593
See also ... 594
Microsoft Teams ... 594
PagerDuty ... 594
Solutions Business Manager ... 595

Slack 595
VictorOps 596
Visual Studio 596
Visual Studio Code 597
Visual Studio for Mac 598
Administration 600
Rules and policy 600
Assess rules 601
Security controls 602
Vulnerability policy 605
Protect rules 609
CVE shields 613
Virtual patches 617
Log enhancers 618
Application exclusions 620
Set compliance policy 623
IP management 624
Library policy 626
Sensitive data masking 627
Notifications 628
Organization 628
Enable Assess 629
Enable Protect 629
Configure 632
Access control for Scan 646
Vulnerability approval 662
View audit log 663
Impersonation 666
System administration 668
Get started 668
Contrast installation 669
Next steps 669
Contrast system requirements 669
Sizing recommendations 670
Download Contrast with curl 671
Download Contrast installer 672
Install 673
Deploy Contrast with a WAR file 676
Distributed MySQL 677
Distributed deployment 679
Run Contrast 682
Credentials 683
Restart Contrast 683
Uninstall 684
Post-installation 684
Post-installation tasks 685
Configure Tomcat 685
JRE 686
Configure HTTPS 686
Customize MySQL 688
Set up a proxy configuration 689
Configure reporting storage 690
Contrast logs 691
Use Redis as a shared cache (on-premises) 693
System updates and upgrades 694

Updates and upgrades ... 694
Upgrade Contrast ... 694
Upgrade agents (on-premises) ... 695
Update your IP address ... 696
Upgrade SCA library data manually ... 696
Upgrade SCA library data automatically ... 697
Update Contrast license ... 698
Manage administration ... 699
Manage multiple organizations ... 699
Add/edit an organization ... 700
Users and permissions ... 702
Add a user ... 702
Add multiple users ... 703
Designate SuperAdmins ... 705
Add access group ... 705
Grant Protect permissions (on-premises) ... 707
Auto-add users ... 708
Credentials ... 710
Impersonate users ... 710
Authentication ... 712
Two-step verification ... 713
Active Directory ... 713
LDAP ... 717
SSO ... 722
HTTPS proxy ... 725
Passwords ... 725
Keys ... 727
System settings ... 727
Steps ... 728
Additional system settings ... 728
General ... 729
Diagnostics ... 729
Licenses ... 730
Score ... 732
Library compliance policy ... 733
Mail ... 733
System maintenance ... 734
Encrypted properties editor ... 734
Manage SSL ... 736
MySQL backups ... 736
Reference ... 739
Glossary ... 739
Roles and permissions ... 742
Application roles ... 743
Organization roles ... 744
System roles ... 746
Application scoring guide ... 747
Library scoring guide ... 748
Log levels ... 749
Events and variables ... 749
Variables ... 750
Regular expressions ... 752
Supported browsers ... 753
Beta Terms and Conditions ... 753
Privacy and data collection ... 753

Get started

Welcome to Contrast!

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

Take a walk through (page 14) 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: • **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.	• Agents (page 29) Supports a variety of programming languages, frameworks, and container technologies that instrument your applications with sensors. • Contrast Assess: (page 21) 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 24) Identifies vulnerabilities in uploaded binary packages by performing a fast and efficient static scan. • Protect: (page 23) 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 22) 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 524) data that includes suggestions on how to fix (page 533) 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 453) 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 553) that let you to integrate Contrast actions and data into developer IDEs, build system, communication tools, and more.

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	Access groups (page 636) 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.
	
	It is useful to plan a group strategy before you add applications to Contrast.
	
	If you do not specify the group in the Contrast configuration file when you first add the 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.
	
	Start by creating or adding a user or application (or both) to an existing group in the Contrast web interface.
	
	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.
	
	`# application:`
	
	`  # Add the name of the application group with which this`
	`  # application should be associated in the Contrast UI.`
	`  # group: NEEDS_TO_BE_SET`

Custom filters	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. You can create custom filters through the use of application metadata. (page 642) You can also apply tags to specific application (page 441) data or vulnerability (page 530) 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. **Example: Application metadata** This example shows how to create free form, custom fields in the Contrast web interface to request application metadata: • **Custom field:** managersInfo **Value:** `"John Doe"` • **Custom field:** businessUnit **Value:** `"NodeGoat Group"` • **Custom field**: officeLocation **Value:**`"New York City"` **Example: Application tags** • **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. **Example: Vulnerability tags** • **Build:** A specific build number • **Version:** A specific release version

Option	Description
Custom data from applications	Session metadata (page 445) lets you identify the source of vulnerabilities in your application. 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. 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. **Example:** 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 `contrast.application.session_metadata` to a set of key-value pairs that identify a branch, a committer, and a repository. `-Dcontrast.application.session_metadata="branchName=build22,committer=Jane,repository=Contrast-Java”`
Custom naming	You have the option of providing customized names for applications (page 441) and servers (page 484) that host the applications. By default, a Contrast agent creates a name based on data it discovers in your code. To specify a custom name, you can use an agent configuration file when you add the application (page 30) to Contrast or set the name in the Contrast web interface after you add an application.

Next steps

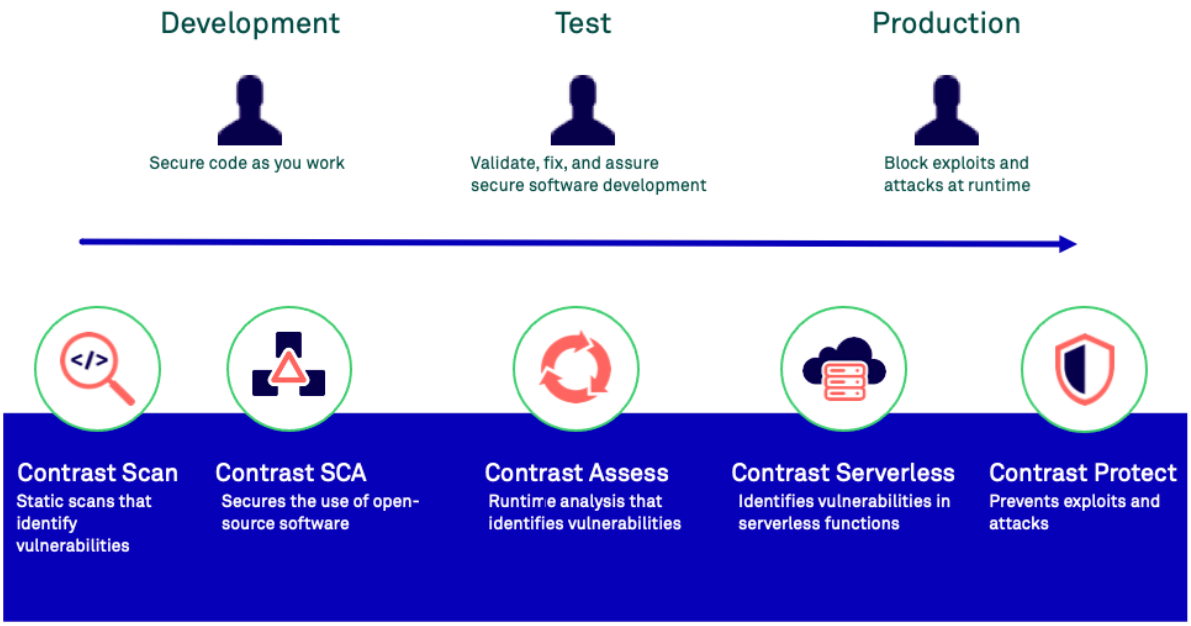
- Get an overview of how Contrast works (page 12)
- Try Contrast for free - Community Edition (CE) (page 27)
- Install and configure a Contrast agent (page 30)
- Install Contrast (on-premises) (page 668)
- Get started with a particular integration (page 553)

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 29) 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 30) 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 custom 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 455) 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 459). 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 24) s a next-generation application security testing solution for serverless-based applications.

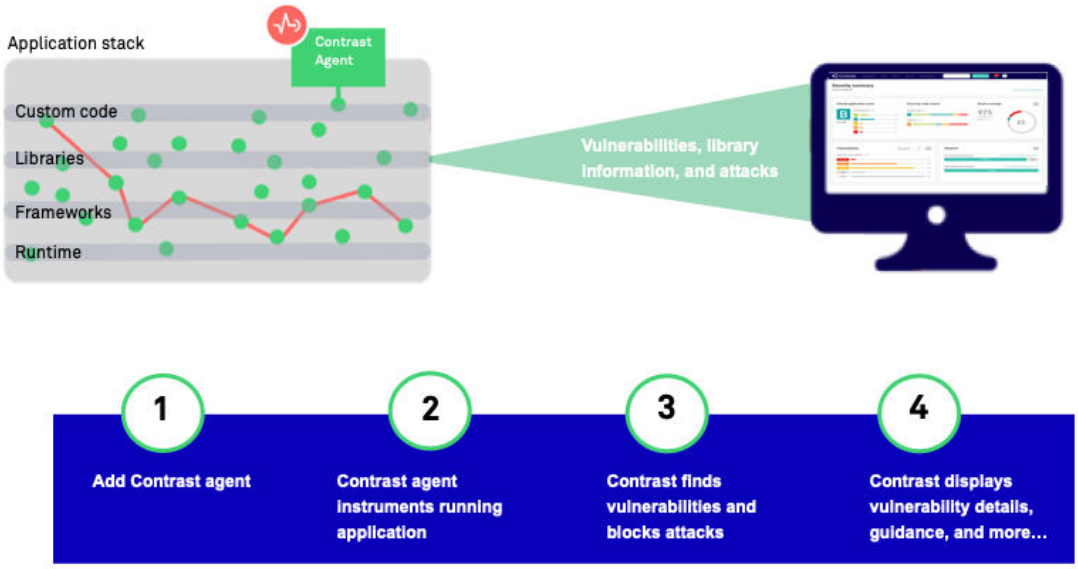
Contrast Serverless Application Security uses cloud-native architecture to map all resources within your environment, while automatically validating and prioritizing the results, eliminating false-positive results and alert fatigue. It uses a ReadOnly access to your AWS account to continuously monitor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

Integrations

Contrast works with several different integrations (page 553) 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.

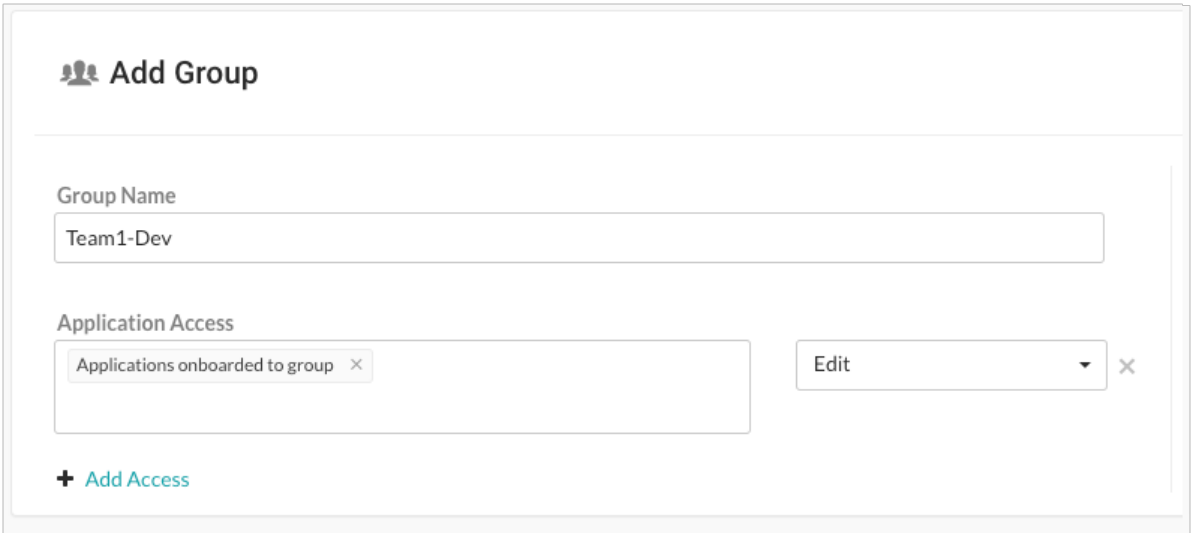
Customize the Contrast environment

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.

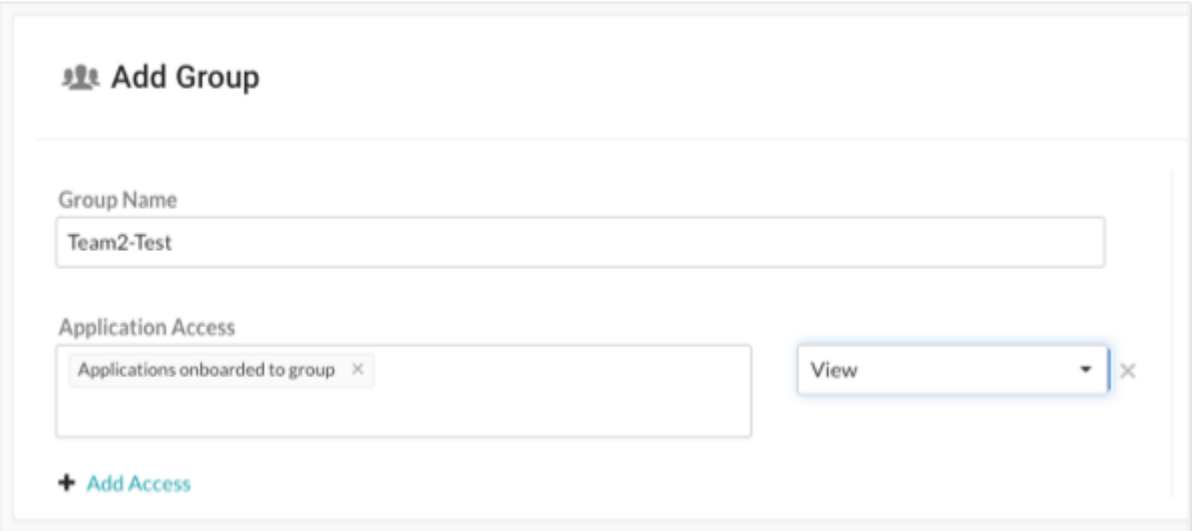
Access groups

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:

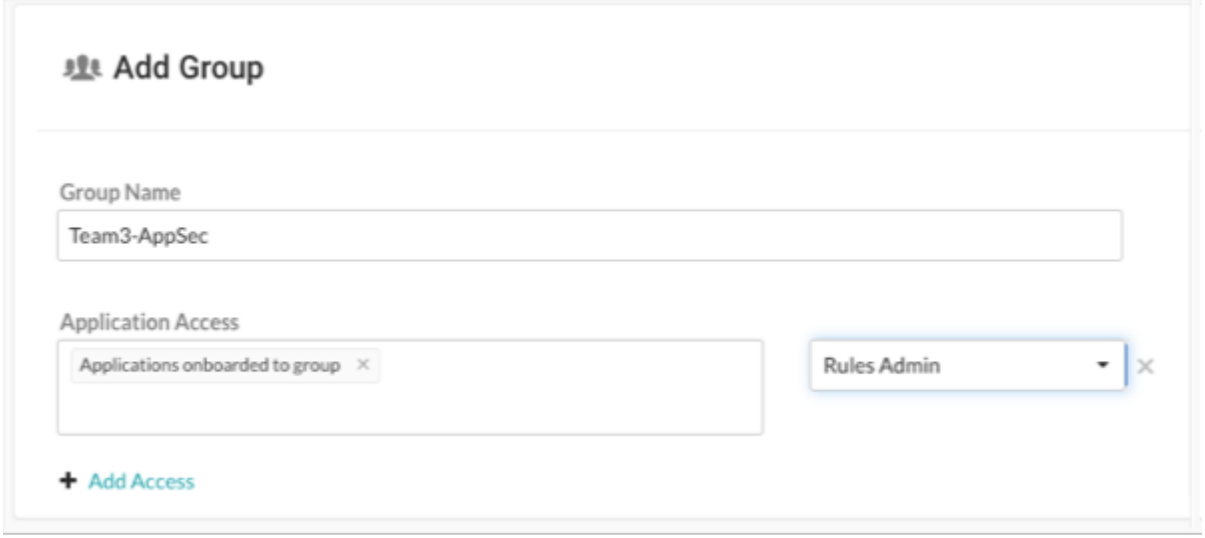


  Developers can remediate findings, add tags, manage vulnerabilities, edit attributes, merge applications, add or delete applications, and create servers.
- **Team2-Test** is for your test team and has these settings:



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”
```

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 54) from the Maven Central repository because you use Maven for your build processes. You also download a YAML configuration file (page 33) 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:A2xxxxxxxxxxxxxxxxxxxxxxxxxxxG9N
    # ********************** REQUIRED **********************
    # Set the service key needed to communicate with the Contrast
    # UI. It is used to calculate the Authorization header.
    service_key:service_key:88xxxxxxxxxxx5Z
    # ********************** 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@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
.
.
.
```

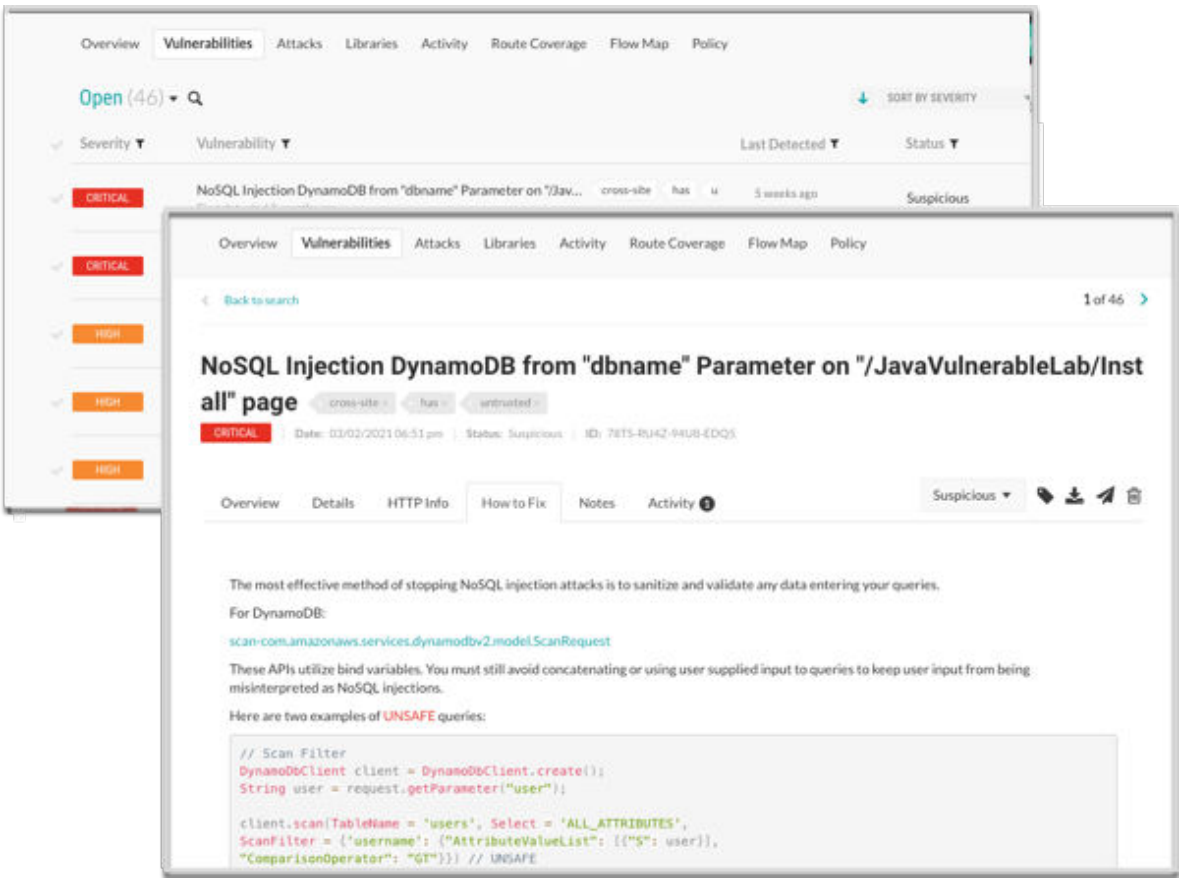
3. For your test environment, you edit a Contrast YAML configuration file with settings similar to these:

```
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:A2xxxxxxxxxxxxxxxxxxxxxxxxxxxG9N
```

```
    # ********************** REQUIRED **********************
    # Set the service key needed to communicate with the Contrast
    # UI. It is used to calculate the Authorization header.
    service_key:service_key:88xxxxxxxxxxx5Z
    # ********************** 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@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.



6. After initial tests, you decide to use the Maven plugin (page 593) 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.

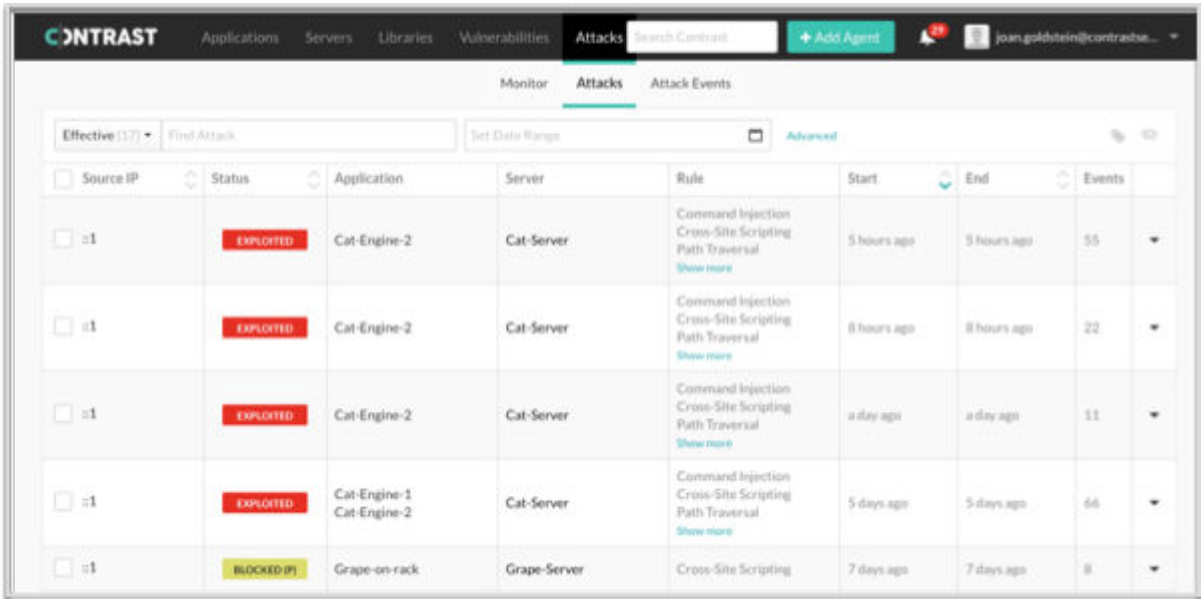
Similar to how you installed and configured an agent for your application in development and test, you need to configure a new configuration file for the production environment that enables Contrast Protect. After you create the new configuration file, you run the application and verify that the Contrast 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:A2xxxxxxxxxxxxxxxxxxxxxxxxxxxxG9N
    # ********************** REQUIRED **********************
    # Set the service key needed to communicate with the Contrast
    # UI. It is used to calculate the Authorization header.
    service_key:service_key:88xxxxxxxxxxx5Z
    # ********************** 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.

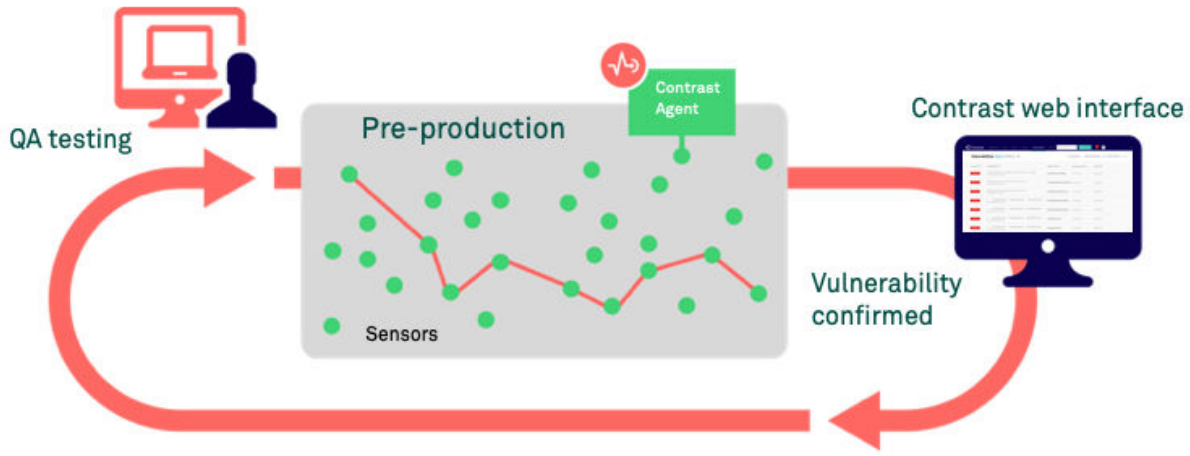


Step 3: Fix code and retest applications

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.

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 30) and enable Assess (page 629), Contrast offers you these features:

- A list of vulnerabilities (page 524) in the application, along with remediation guidance.
- Application scores (page 747) to gauge the security of an application at a glance.
- Route coverage (page 448) that detects possible routes by associating vulnerabilities with the originating web request.
- Flow maps (page 453) that provide insight into the architecture of the running application.
- Compliance and policy reporting (page 547).

Customization

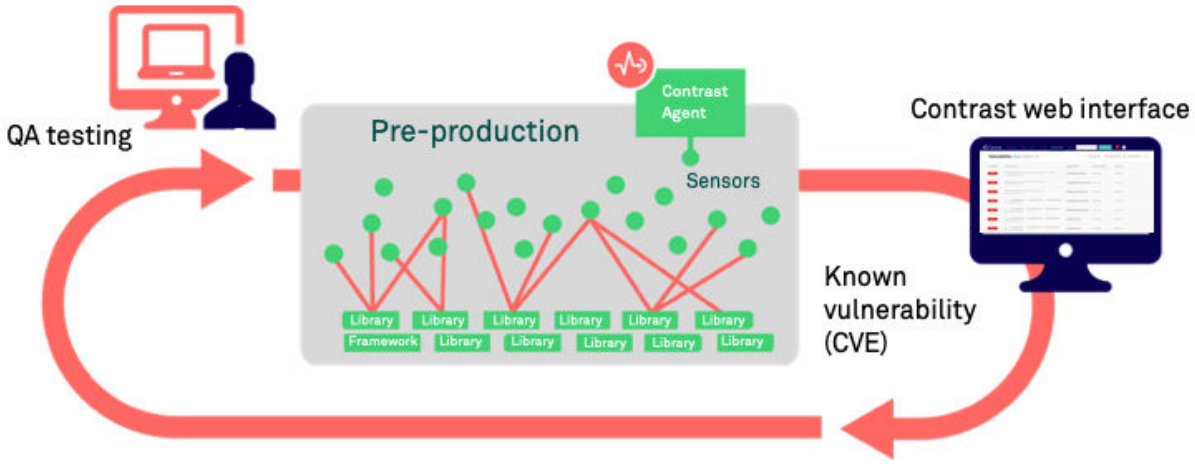
To customize Assess for your needs, you have the option of configuring these types of policies:

- Assess rules (page 601) that you can enable or disable to fine tune the detection capabilities of Assess.
- Security controls (page 602) 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 495) 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 623) 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 626).
  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 518) collects is used to display a dependency tree (page 495) 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.

Contrast data

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

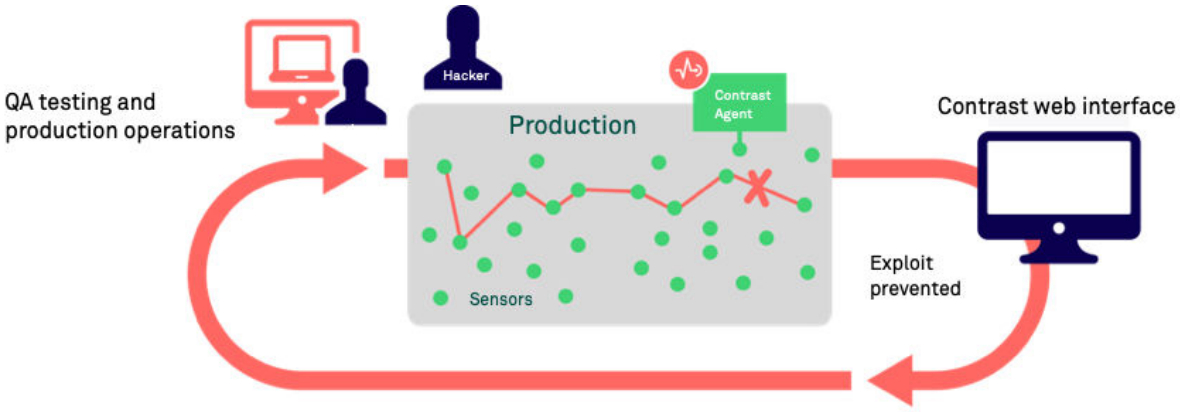
- Library usage analysis to identify whether vulnerable components are actually used by the application
- Library version identification and guidance on the latest version
- Comprehensive vulnerabilities coverage
- Portfolio wide, real-time reporting of open-source components

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 52), .NET (page 126), .NET Core (page 180), Node.js (page 229), Ruby (page 353), and Python (page 294), 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 609)** Set applications to monitor for attacks.
- **CVE shields:** (page 613) Specify how to CVE shields to block vulnerabilities.
- **Virtual Patches: (page 617)** Define custom defenses against specific vulnerabilities.
- **Log Enhancers: (page 618)** Provide additional instrumentation instructions.

- **IP Management: (page 624)** Manage a denylist and allowlist (trusted hosts).

Scan

Contrast Scan (page 455) 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

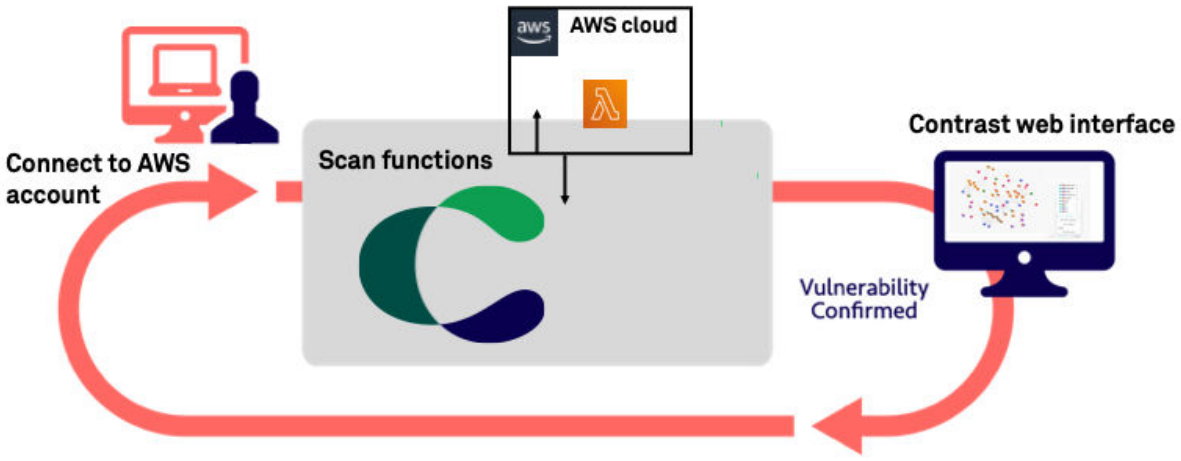
Scan supported languages (page 457)

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 499)

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.

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 or .NET Core application to Contrast.

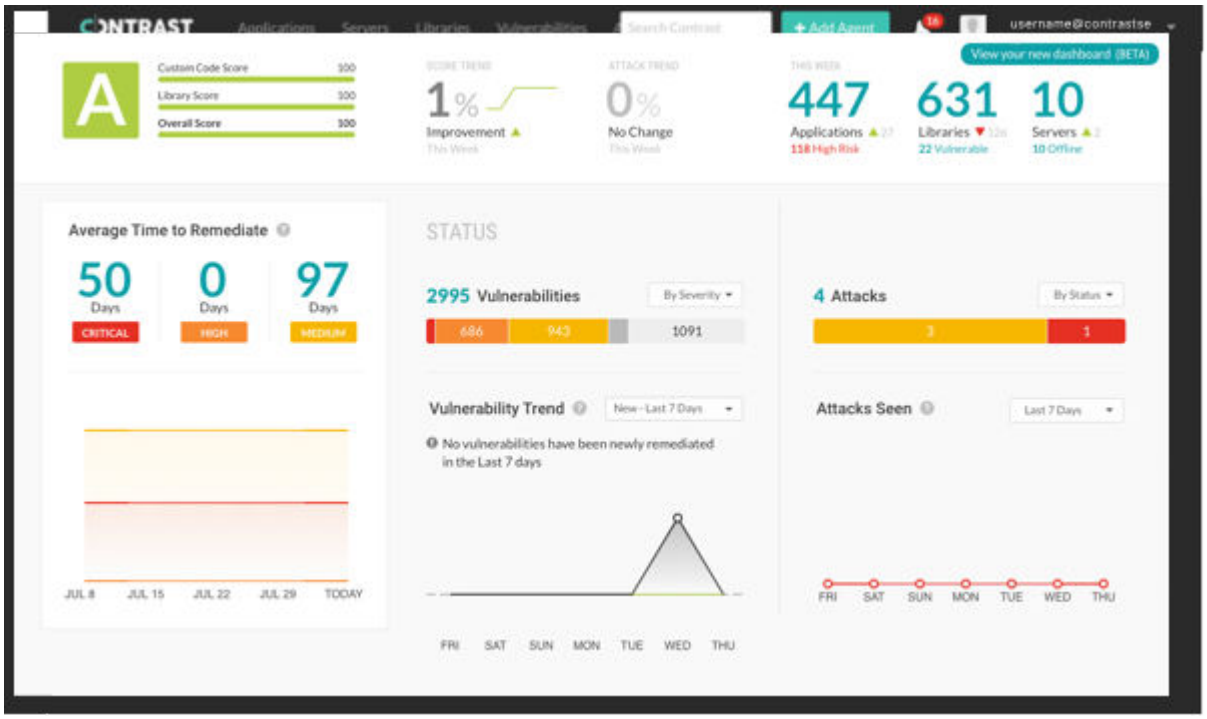
Community Edition features

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 182)
- Install the Java agent (page 54)

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 30), 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 533) 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:

- Go (page 409) instruments Go web applications for library support and vulnerability reporting.
- Java (page 52) instruments Java web applications and web APIs running on your container.
- .NET Framework (page 126) instruments .NET web applications and APIs running on IIS.
- .NET Core (page 180) instruments applications and APIs running in the .NET Core runtime.
- Node.js (page 229) instruments Node.js web applications and APIs.
- Ruby (page 353) instruments Ruby on Rails web applications.
- PHP (page 263) analyzes PHP web applications at runtime for library usage and vulnerability detection.
- Python (page 294) instruments Django, Flask and Pyramid web applications.

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 and configure an agent

To instrument an application with Contrast you need to install and configure the agent that corresponds to the language your application uses. Most agents can be downloaded from a package manager or repository.

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

Installation

Depending on your situation, you may install the agent directly or with the help of integrations (page 553) that work with Contrast.

Consider your goals for Contrast and consult specific installation instructions for the agent you want to use:

- Go (page 412)
- Java (page 54)
- .NET Framework (page 128)
- .NET Core (page 182)
- Node.js (page 232)
- PHP (page 264)
- Python (page 296)
- Ruby (page 355)

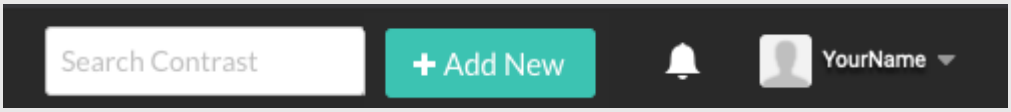
Configuration

When you install an agent, you must configure it so that it recognizes your application and can communicate information back to Contrast. At a minimum, you need to supply four basic agent keys (find them here) (page 32):

- **api.api_key:** Your organization's API key
- **api.user_name:** Contrast user account (in most cases your login ID)
- **api.service_key:** Contrast user account service key
- **api.url:** Address of the Contrast installation you would like your agent to report to. Defaults to: https://app.contrastsecurity.com

TIP

You can download a YAML configuration file that is pre-populated with your organization keys. Select **Add new** in the Contrast web interface and choose your application language to find a download link. And you can also configure the file with the Contrast agent configuration editor (page 34) via the **Open YAML Editor** link.



You can further configure by using environment variables (page 51), a YAML configuration file (page 33), command line or other methods native to the language and tools you are using. Configuration follows this order of precedence (page 31).

Consider your goals for Contrast and consult specific configuration instructions for the agent you want to use:

- Go (page 415)
- Java (page 70)
- .NET Framework (page 140)
- .NET Core (page 197)
- Node.js (page 239)
- PHP (page 266)
- Python (page 298)
- Ruby (page 357)

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.
   For example: `CONTRAST__ENABLE`
4. An application-specific configuration file (.NET Framework only).
   For example: *web.config (page 141)*
5. Configuration values in a YAML file are pulled from the first file found in the following order:
   a. A YAML file indicated by the user
      For example:
      - Java: the `contrast.config.path`system property (page 70)
      - Node.js: the `--configFile`command line flag
        Node.js: the `--configFile`command line flag
      - Any agent: the `CONTRAST_CONFIG_PATH`environment variable (page 51)
   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 Java, .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.

Find the agent keys

IMPORTANT

If you download the YAML configuration file from Contrast (select **Add new** in the top right) the file is pre-populated with your agent keys.

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

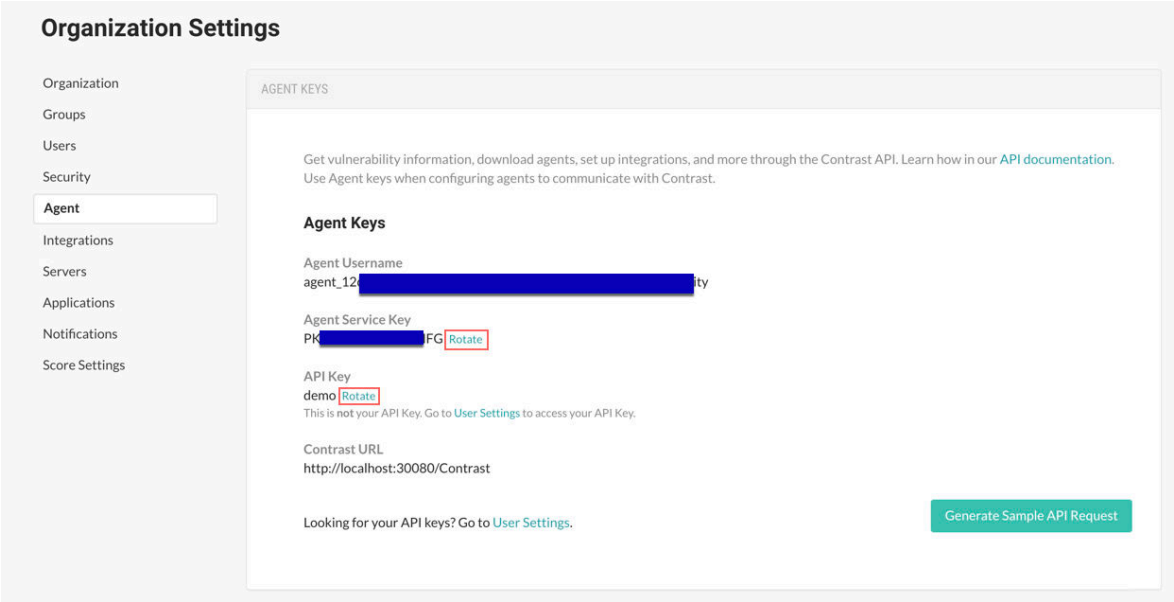
Agent keys are common to all agents in an organization. They are values that represent and identify the agents as well as the organization being accessed.

All these keys are required when installing agents:

- Agent username
- 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** to see the values for agent keys.



The Contrast URL is *https://app.contrastsecurity.com/Contrast*, or the URL of your on-premises or private cloud instance.

NOTE
If you don't see the agent username or service key on this page, it may mean that a license has not been applied to your organization. Contact Support for help with this.

3. 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.

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 31).

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 own configuration file, you will need to add these keys (page 32) yourself.

The minimum required *contrast_security.yaml* content for all agents should look like this:

```
api:
    url: https://app.contrastsecurity.com
    user_name: contrast_user
    api_key: demo
    service_key: demo
```

TIP

- Use the Contrast agent configuration editor (page 34) 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:

- Go (page 416)
- Java (page 71)
- .NET Framework (page 142)
- .NET Core (page 198)

- Node.js (page 240)
- PHP (page 266)
- Python (page 298)
- Ruby (page 358)

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 (Beta)

IMPORTANT
This feature is in beta. Beta status means the feature might change or act unexpectedly. By using this feature, you agree to the Contrast Beta Terms and Conditions (page 753).

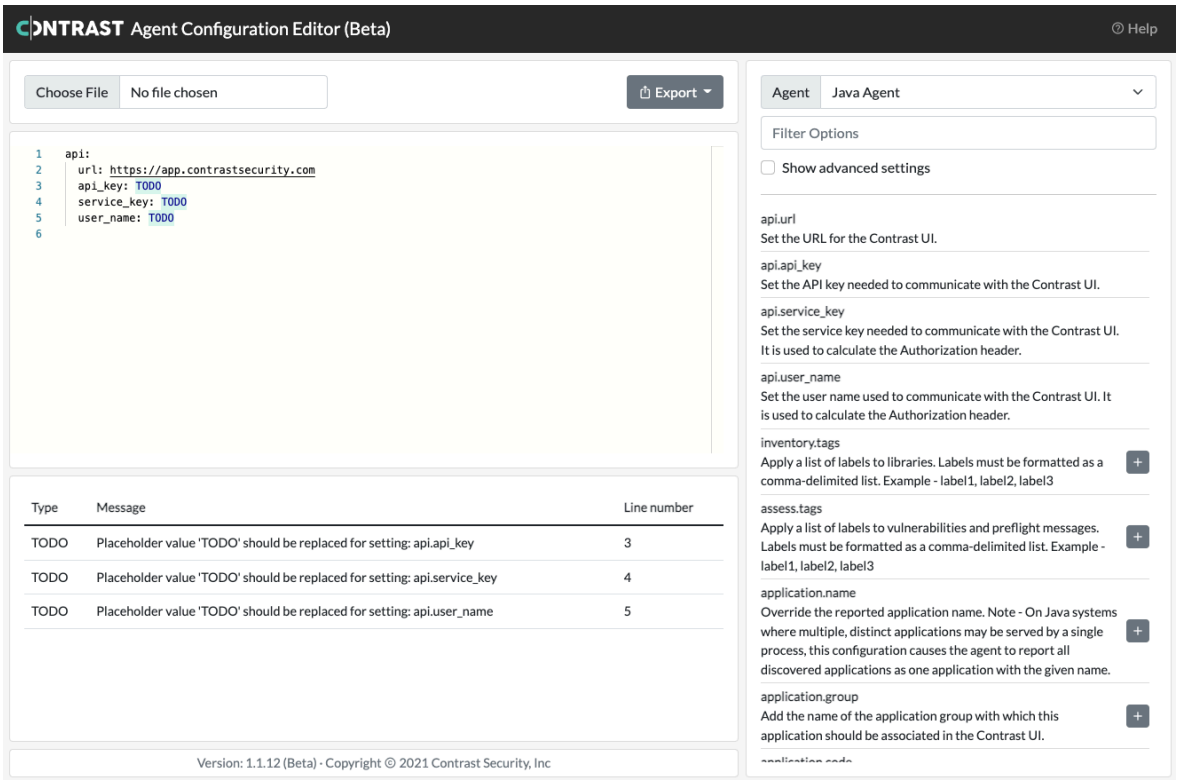
The Contrast agent configuration editor is a web application that can be used to validate and generate configuration for Contrast agents.

Before you begin

- Use this editor to help edit, validate, and generate configuration for Contrast agents.
- If you already have a YAML file, you can open it in the editor by selecting **Choose File**.
- 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 agent configuration editor in your browser.



2. Either import an existing YAML file by selecting Choose File, or enter 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 the **agent language** in the drop-down 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.

> **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 agent configuration file either as a YAML or environment variables.

NOTE
At this time, the Contrast agent 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.

Agent Operator (Kubernetes operator)

IMPORTANT
This feature is in beta. Beta status means the feature might change or act unexpectedly. By using this feature, you agree to the Contrast Beta Terms and Conditions (page 753).

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.

To get started, install the agent operator (page 37) or use the agent operator walkthrough (page 38) for a full example.

The operator is configured using declarative Kubernetes native resource types. Resources types are documented in the agent operator configuration (page 45) section.

NOTE
The Contrast Agent Operator is an open source project. You can review the code and contribute to its development here.

See also

- Agent Operator supported technologies (page 36)
- Agent Operator telemetry (page 50)
- Agent Operator minimum configuration (page 42)

Supported technologies for Agent Operator

Kubernetes / OpenShift support

Kubernetes version	OpenShift version	Supported	End-of-support
v1.23	v4.10	Yes	2023-02-28
v1.22	v4.9	Yes	2022-10-28
v1.21	v4.8	Yes	2022-06-28

- 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 hosts and will refuse to be scheduled onto incompatible nodes. Additionally, the operator only supports injecting workloads running on Linux amd64 hosts, even if the Contrast Agent supports additional platforms. Contact Contrast Support if Kubernetes on Windows or arm64 support is desired.

Agent types

Agent	Agent type	Support status	Compatibility notes
.NET Core	dotnet-core	Supported	Supported .NET Core technologies (page 181)
Java	java	Supported	Supported Java technologies (page 52)
NodeJS	nodejs	Supported	Supported NodeJS technologies (page 230)
PHP	php	Beta	Supported PHP technologies (page 263)

NOTE

- Injection of a PHP application is in beta. Beta status means the feature might change or act unexpectedly. By using this feature, you agree to the Contrast Beta Terms and Conditions (page 753).
- Injection of the NodeJS 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 NodeJS agent during compilation then injection during runtime by the operator should be disabled. See the rewriter CLI (page 259) for more information.

Install the Agent Operator

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.

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.

Steps

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.

> **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.

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
```

3. When the wait command succeeds the operator is ready to be configured (page 42).

See also

- Agent Operator walkthrough (page 38)
- Agent Operator minimum configuration (page 42)
- Agent Operator telemetry (page 50)

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.

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. 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

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 called `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

secret/default-agent-connection-secret created
```

NOTE

Replace `TODO` with the equivalent values for your Contrast server instance. Find the agent keys (page 32) 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
```

```
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 .NET Core Agent into the ASP.NET Core sample application using a Deployment workload.

First, deploy the ASP.NET Core 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: hello-world-app
  namespace: default
  labels:
    arbitrary-label: arbitrary-value
spec:
  selector:
    matchLabels:
      app: hello-world-app
  template:
    metadata:
      labels:
        app: hello-world-app
    spec:
      containers:
        - image: mcr.microsoft.com/dotnet/samples:aspnetapp
          name: hello-world-app
EOF

deployment.apps/hello-world-app created
```

After waiting for cluster convergence, the deployed workload should be ready in the `Running` status.

```
$% kubectl -n default get pods

NAME                               READY   STATUS    RESTARTS   AGE
hello-world-app-7479d5ff96-p28zx   1/1     Running   0          19s
```

Next, the operator can be configured to inject the .NET Core 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: injector-for-hello-world
  namespace: default
spec:
  type: dotnet-core
  selector:
    labels:
      - name: arbitrary-label
        value: arbitrary-value
EOF

agentinjector.agents.contrastsecurity.com/injector-for-hello-world created
```

Checking the logs of the `hello-world-app` Pod shows that the Contrast .NET Core agent is now instrumenting the application.

```
% kubectl -n default logs Deployment/hello-world-app

Defaulted container "hello-world-app" out of: hello-world-app, contrast-
init (init)
warn: \
Microsoft.AspNetCore.DataProtection.Repositories.FileSystemXmlRepository[60]
      Storing keys in a directory '/root/.aspnet/DataProtection-Keys' that \
may not be persisted outside of the container. Protected data will be \
unavailable when container is destroyed.
warn: Microsoft.AspNetCore.DataProtection.KeyManagement.XmlKeyManager[35]
      No XML encryptor configured. Key \
{0ad20893-267f-4635-99b0-1ee74bccbc8b} may be persisted to storage in \
unencrypted form.
            ___
       _.-|   |          |\__/,|   (`\     Contrast .NET Core Agent 2.1.13.0
      [   |   |          |o o  |__ _) )
       `-.|___|        _.( T   )  `  /     Contrast UI: https://
app.contrastsecurity.com
         .--'-`-.     _((_ `^--' /_<  \     Mode:        Assess & Protect
       .+|______|__.-||__)`-'((((/  ((((/

info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://[::]:80
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Production
info: Microsoft.Hosting.Lifetime[0]
      Content root path: /app/
```

Step 4: Uninstall the operator (optional)

To restore the original state of the cluster, first remove existing AgentInjectors.

```
% kubectl -n default delete agentinjector injector-for-hello-world

agentinjector.agents.contrastsecurity.com "injector-for-hello-world" 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

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 37)
- Agent Operator minimum configuration (page 42)
- Agent Operator configuration (page 45)

Agent Operator minimum configuration

All configuration of the operator is handled through the use of Kubernetes native configuration entities defined by custom resource definitions (CRDs). The CRDs are deployed with the operator and define how to interact with the operator's configuration entities.

Tooling such as Visual Studio Code's Kubernetes extension can aid in creating syntactically correct entities in your cluster.

The full schema is documented in the Agent Operator configuration (page 45). This section only covers the minimal setup required and may not cover all situations.

Minimum configuration

For a minimum setup, 3 manifests are required.

1. First, a standard Kubernetes Secret contains the necessary connection keys to authenticate to your Contrast server instance. The Secret must be deployed into the same namespace as the ClusterAgentConnection entity. You can find your agent keys under find the agent keys (page 32).

   ```
   apiVersion: v1
   kind: Secret
   metadata:
     name: default-agent-connection-secret
     namespace: contrast-agent-operator
   type: Opaque
   stringData:
     apiKey: TODO
   ```

```
  serviceKey: TODO
  userName: TODO
```

2. Second, a ClusterAgentConnection configuration entity. The ClusterAgentConnection provides the default connection settings for agents within the cluster and maps to the above mentioned Secret containing connection authentication keys. For security, ClusterAgentConnection entities must be deployed into the same namespace as the operator to be used. This example assumes that the default namespace `contrast-agent-operator` hasn't been customized.

```
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
```

3. Finally, a AgentInjector configuration entity. The AgentInjector selects workloads eligible for automatic injection using workload labels e.g. `metadata.labels` within the namespace in which the AgentInjector is deployed.

```
apiVersion: agents.contrastsecurity.com/v1beta1
kind: AgentInjector
metadata:
  name: dotnet-hello-world
  namespace: default
spec:
  type: dotnet-core
  selector:
    labels:
      - name: app
        value: dotnet-hello-world
```

In this example manifest, the Contrast Agent Operator will automatically inject the .NET Contrast agent into workloads (e.g. Deployments, DeploymentConfigs, etc.) that have the label `app=dotnet-hello-world` in the namespace default.

See also

- Agent Operator configuration (page 45)
- Agent Operator supported technologies (page 36)

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 45)

- Operator supported technologies (page 36)

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.

```
kubectl delete crd agentconfigurations.agents.contrastsecurity.com
kubectl delete crd agentconnections.agents.contrastsecurity.com
kubectl delete crd agentinjectors.agents.contrastsecurity.com
kubectl delete crd clusteragentconfigurations.agents.contrastsecurity.com
kubectl 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.

```
kubectl 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.

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.

Property	Type	Required	Default value	Description
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', 'php'].

Property	Type	Required	Default value	Description
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
PHP	php

Types are further documented in Operator supported technologies (page 36).

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 37)
- Agent Operator supported technologies (page 36)
- Agent operator telemetry (page 50)

.NET Core chaining support

The .NET Core Agent, paired with the Contrast Agent Operator (page 36), 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.

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.

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
contrast.server.name	`server:` `  name:`	CONTRAST__SERVER__NAME
contrast.api.api_key	`api:` `  api_key:`	CONTRAST__API__API_KEY

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

- Go (page 416)
- Java (page 71)
- .NET Framework (page 142)

- .NET Core (page 198)
- Node.js (page 240)
- PHP (page 266)
- Python (page 298)
- Ruby (page 358)

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 54).

Supported technologies for Java (Kotlin, Scala) agent

Java

Technology	Supported versions	Notes
Java runtime	• IBM 8 • Oracle 8. *Versions 11+ follow our OpenJDK support. • OpenJDK 8, 11, 12, 13, 14, 15, 16, 17	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.
Java runtime for Legacy Java agent (page 125) Use with Java agent 3.x only	• IBM 6, 7 • Oracle 6, 7 • OpenJDK 6, 7	**See also** • Support-Bulletin-End-of-Support-of-Java-6-and-7 • FAQ: End of Support for Java 6 and 7
Application servers	• GlassFish 4 • 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 • Play 2.4 • Resin 4 • Tomcat 5, 6, 7, 8, 9 • Vert.X 3.1.0 • WebLogic 10, 11g, 12c • WebSphere* 8.5, 9.0 • WildFly 10, 11, 14, 18	* Contrast offers limited support for zSeries and AIX environments. Customers using WebSphere on SPARC Solaris require version 8.5.5.11.
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	• DB2 • DynamoDB • MySQL • Oracle • PostgreSQL • SQL Server • SQLite JDBC drivers	

Technology	Supported versions	Notes
Message services	• Message services: JMS 2.0 • IBM MQ 9.x • Spring JMS 2.x	• Agent version: Java 4.7.0 and later • Contrast version: 3.9.9 and later **NOTE** If you are interested in testing message services in Contrast, contact your customer success representative or Customer support.
Other Java technologies	• 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) • Hibernate • 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 • Seam • Spring, Spring Boot, Spring AOP • Struts, Struts 2 • Wicket • XStream, Jackson (JSON/XML) • Xerces, JAXB, nu.xom	*If you are using Glowroot, the Contrast Java Agent jar should be included and loaded prior to the Glowroot jar.

Kotlin

Technology	Supported versions
Contrast agent	
Java Run Time	JDK 8 and up
Kotlin version	1.5
SpringBoot	

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 67) before you deploy the agent.

System requirements for the Java agent

Before installing the Java agent confirm you can meet the following requirements:

- one
- two
- three

Requirement	Recommended	Notes
Framework		
CPU		
Memory		

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.

Quick start

Just want to try out the Java agent and see how it works? Check out this Java Quick Start Guide (page 68).

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 54)
- Debian (page 55)
- RPM (page 56)

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 593)
- Gradle plugin (page 579)
- Jenkins plugin (page 580)

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.)
2. Configure the agent (page 70). You can create or download a YAML configuration file (page 33). You must provide Contrast connection parameters using these agent keys (page 32).
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 31) 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 64)
- JBoss/Wildfly (page 63)
- Tomcat (page 65)
- WebLogic (page 66)
- WebSphere (page 67)

You can also install using a container (page 57), like Docker.

TIP
Check the Contrast Support Portal for more information about other compatible ways to install the agent using tools like Pivotal Cloud Foundry (now VMware Tanzu).

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 70). You can create or download a YAML configuration file (page 33). You must provide Contrast connection parameters using these agent keys (page 32).

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 31) 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 64)
- JBoss/Wildfly (page 63)
- Tomcat (page 65)
- WebLogic (page 66)
- WebSphere (page 67)

You can also install using a container (page 57), like Docker.

TIP

Check the Contrast Support Portal for more information about other compatible ways to install the agent using tools like Pivotal Cloud Foundry (now VMware Tanzu).

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 70). You can create or download a YAML configuration file (page 33). You must provide Contrast connection parameters using these agent keys (page 32).
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 31) 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 64)
- JBoss/Wildfly (page 63)
- Tomcat (page 65)
- WebLogic (page 66)
- WebSphere (page 67)

You can also install using a container (page 57), like Docker.

TIP

Check the Contrast Support Portal for more information about other compatible ways to install the agent using tools like Pivotal Cloud Foundry (now VMware Tanzu).

Install the Java agent using a container

Before you begin

This topic provides general guidance for installing the Contrast Java 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: Add the agent to base application image

If you add the agent to a base image, you can make a single image change and Contrast will be available to all applications using that base image. Also, this way, updates will depend on the base image update. To do this:

1. Add the Contrast agent and basic configuration (page 70) to a Docker base image, but don't enable it. The preferred location is `/opt/contrast`. For example:

   ```
   FROM BASE_IMAGE
   ARG CONTRAST_AGENT_VERSION
   ADD https://repo1.maven.org/maven2/com/contrastsecurity/contrast-agent/
   $CONTRAST_AGENT_VERSION/contrast-agent-$CONTRAST_AGENT_VERSION.jar
   /opt/contrast/contrast.jar
   ```

   Optionally, you can change the URL to download agents from an internal repository
   You can pass a specific agent version at build time by replacing `<YourAgentVersion>` with the version number you want to download, as shown in this example:

   ```
   docker build --build-arg CONTRAST_AGENT_VERSION=<YourAgentVersion> -
   t image_with_contrast:tag .
   ```

   TIP
   If you want more flexibility to use any version of the Java agent, and to avoid automatic updates, apply the ADD instruction directly to an application's Docker image.

2. To enable Contrast in the application's Docker image or container, use JVM parameters (page 59).

Step 2: Configure the agent

Configuration of the Java agent follows this order of precedence. When installing into a container:

- Use a **YAML configuration file** for common configuration so it can be placed in the base image. For example, common configuration might include redirecting logging to console output, proxy configuration, or performance tuning.
  Here is a sample YAML configuration file:

  ```
  agent:
    java:
      scan_all_classes: false
      scan_all_code_sources: false
    logger:
      stdout: true
  ```

  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	`-Dcontrast.application.metadata`	CONTRAST__AP

Contrast configuration	Function	Java system property	E
Application session metadata	Send application details like build number, version, GIT hash, and other session metadata (page 445).	`-Dcontrast.application.session_metadata`	CONTRAST__AP
Application group	Specify the application access group for this application during onboarding. Create these groups (page 636) in Contrast first.	`-Dcontrast.application.group`	CONTRAST__AP
Server environment	Specify in which environments the application is running: Development, QA and Production.	`-Dcontrast.server.environment`	CONTRAST__SE

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 58) and configure (page 58) the agent in a base image, run the image.

For the agent to send data to Contrast, it needs agent authentication keys (page 32). 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 36)

Contrast Support Portal AWS Fargate and Contrast agents

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

curl --fail --silent --location "https://
repository.sonatype.org/service/local/artifact/maven/redirect?r=central-
proxy&g=com.contrastsecurity&a=contrast-agent&v=LATEST" -o /tmp/
$CONTRAST_FILE_NAME
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 32).
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

Run your Scala Play application

To run the Scala Play application, you must enable three properties:

- `contrast.agent.java.enable_scala_support`
- `contrast.agent.java.enable_akka_support`
- `contrast.agent.java.enable_play_support`

To configure the agent, use the JAVA_OPTS environment variable to attach and configure the agent. (Alternatively, if you have a standalone fat JAR, you can use system properties.)

Enable these properties (with YAML, command line or environment variables):

```
-Dcontrast.agent.java.enable_scala_support=true
-Dcontrast.agent.java.enable_akka_support=true
-Dcontrast.agent.java.enable_play_support=true
```

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 63)
- Jetty (page 64)
- Tomcat (page 65)
- Weblogic (page 66)
- WebSphere (page 67)

See also

- Install the Java agent (page 54)
- Supported technologies (page 52)
- Configure the Java agent (page 70)

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 54)
   - Debian (page 55)
   - RPM (page 56)
2. You can either run JBoss from a BAT file, or in domain mode.
   - **BAT file:** If you run JBoss from *domain.bat*, *standalone.bat*, or *run.bat* with a *.conf* 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 52) file, and use the JBoss server directory for your environment. Then add this line to the end of your *.conf* 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 52) 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 124). 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 52) 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 54)
   - Debian (page 55)
   - RPM (page 56)
2. On your Jetty environment, replace `<YourContrastJarPath>` with the path to your Contrast JAR (page 52) 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 52) 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 54)
- Debian (page 55)
- RPM (page 56)

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 52) 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 54)
- Debian (page 55)
- RPM (page 56)

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 52) 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 52) 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 *catalina.policy* file). Replace `<YourContrastJarPath>` with the path to your Contrast JAR (page 52) 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 54)
- Debian (page 55)
- RPM (page 56)

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 52) file. For example:

```
<WebsphereDirectory>\AppServer\profiles\AppSrv01\config\cells\<CellName>\nod
es\<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 52) file. For example:

```
grant codeBase "file:<YourContrastJarPath>" {
  permission java.security.AllPermission;
};
```

2. Append the `contrast.policy` file to the `$WEBSPHERE_HOME/AppServer/profiles/AppSrv01/properties/server.policy` .
3. Enable the security manager with the `wsadmin` tool:
   - **Jacl:**`$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 553) 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 52).

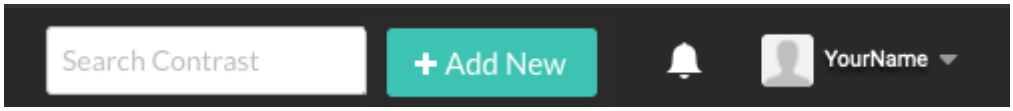
You will also need access to a command line interface (with a chosen directory for downloading the agent) and your organization's instance of Contrast. If you don't already have Contrast, you can sign up for the Community Edition (page 27) for free.

Install

1. Use this command to download the agent JAR file from Maven Central. From your command line interface, enter this curl request to pull down the latest version of the agent to your current directory.

   ```
   curl -
   L 'https://repository.sonatype.org/service/local/artifact/maven/redirect?
   r=central-proxy&g=com.contrastsecurity&a=contrast-agent&v=LATEST' -
   o contrast.jar
   ```

   Once complete, you should see the `contrast.jar` file in your current directory. (For example: `./contrast.jar`)
2. Download the YAML configuration file from the agent wizard (if you haven't already done so). To do this, in the Contrast application, select **Add new**.

   Select **Java** as your language, then select **Download YAML configuration file**. The file will download locally and will contain the specific agent keys for your organization that will connect your application to Contrast.
3. To configure the agent, open the YAML configuration file in an editor.

   ```
   api:
     url: https://xxx.contrastsecurity.com/Contrast
     api_key: A2xxxxxxxxxxxxxxxxxxxxxxxxxxxxG9N
     service_key: 88xxxxxxxxxxx5Z
     user_name: agent_xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx@OrgName
   ```

4. Tell the agent where to find your YAML configuration file by entering this command in your command line interface.

   ```
   java -javaagent:./contrast.jar -
   Dcontrast.config.path=contrast_security.yaml -jar <ApplicationJarPath>
   ```

   Be sure to replace `<ApplicationJarPath>` with the path to your application. For example: `./MyApplication.jar`
5. 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 30) for all agents uses this order of precedence (page 31).

You can configure the Java agent using:

- Java system properties
- Environment variables (page 51)
- Java YAML template

TIP
Use the Contrast agent configuration editor (page 34) 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 123)
- Java 9 Modules (page 124)
- Java 2 Security (page 124)
- Integrations: (page 553) 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 52), 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 33).)

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

  # 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:

  # Set the amount of time to run the agent before shutting down itself
  # (in milliseconds). A negative value disables scheduled shutdown.
```

```
  # shutdown_time_ms: NEEDS_TO_BE_SET

  # ================================================================================
====
  # 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.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. 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 at which to send 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.
  # 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

  # 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.
    # 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:

  # 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: false

  # ================================================================================
====
  # protect.rules
  # Use the following properties to set simple rule configurations.
  # ================================================================================
====
```

```
  # rules:

    # Define a list of Protect rules to disable in the agent.
    # 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: true

      # 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: true

      # 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: true

      # 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: true

      # 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.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.csrf
    # Use the following properties to configure how
    # the cross-site request forgery rule works.
    # ================================================================================
    # csrf:

      # Set the mode of the rule. Value options are
      # `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 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
```

```
  # Override the reported server environment. Valid
  # values include `QA`, `PRODUCTION` and `DEVELOPMENT`.
  # environment: development

  # 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

  # 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:

  # Set the amount of time to run the agent before shutting down itself
  # (in milliseconds). A negative value disables scheduled shutdown.
  # shutdown_time_ms: NEEDS_TO_BE_SET

  # ================================================================================
```

```
====
  # 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.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. 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 at which to send 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.
  # 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

  # 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.
    # 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:

  # 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: false

  # ================================================================================
====
  # protect.rules
  # Use the following properties to set simple rule configurations.
  # ================================================================================
====
  # rules:

    # Define a list of Protect rules to disable in the agent.
```

```
    # 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: true

      # 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: true

      # 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: true

      # 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: true

      # 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.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.csrf
    # Use the following properties to configure how
    # the cross-site request forgery rule works.
    # ========================================================================
====
    # csrf:

      # Set the mode of the rule. Value options are
      # `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 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

  # Override the reported server environment. Valid
  # values include `QA`, `PRODUCTION` and `DEVELOPMENT`.
```

```
  # environment: development

  # 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
```

Java system properties

Substitute <YourContrastJarPath> with the path to your Contrast JAR (page 52), 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 33).)

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

  # 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:

  # Set the amount of time to run the agent before shutting down itself
  # (in milliseconds). A negative value disables scheduled shutdown.
  # shutdown_time_ms: NEEDS_TO_BE_SET

  # ================================================================================
====
  # 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.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. 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 at which to send 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.
  # 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

  # 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.
    # 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:

  # 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: false

  # ================================================================================
====
  # protect.rules
  # Use the following properties to set simple rule configurations.
  # ================================================================================
====
  # rules:

    # Define a list of Protect rules to disable in the agent.
    # 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: true

      # 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: true

      # 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: true

      # 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: true

      # 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.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.csrf
    # Use the following properties to configure how
    # the cross-site request forgery rule works.
    # ================================================================================
====
    # csrf:

      # Set the mode of the rule. Value options are
      # `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 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

  # Override the reported server environment. Valid
  # values include `QA`, `PRODUCTION` and `DEVELOPMENT`.
  # environment: development

  # 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

  # 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:

  # Set the amount of time to run the agent before shutting down itself
  # (in milliseconds). A negative value disables scheduled shutdown.
  # shutdown_time_ms: NEEDS_TO_BE_SET

  # ============================================================================
====
  # 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.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. 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 at which to send 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.
  # 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

  # 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.
    # 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:
```

```
  # 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: false

  # ================================================================================
====
  # protect.rules
  # Use the following properties to set simple rule configurations.
  # ================================================================================
====
  # rules:

    # Define a list of Protect rules to disable in the agent.
    # 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: true

      # 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: true
```

```
      # 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: true

      # 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: true

      # 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.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.csrf
    # Use the following properties to configure how
    # the cross-site request forgery rule works.
    # ============================================================================
====
    # csrf:

      # Set the mode of the rule. Value options are
      # `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 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

  # Override the reported server environment. Valid
  # values include `QA`, `PRODUCTION` and `DEVELOPMENT`.
  # environment: development

  # 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
```

Configure the Java agent for standalone applications

NOTE

Standalone application configuration is not needed with the Java agent 4.X.

If you are still using 3.X, check the Java agent legacy (page 125) documentation.

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 52), 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 64)
- Tomcat (page 65)
- WebLogic (page 66)
- WebSphere (page 67)
- WildFly (page 63)

Legacy Java agent

Since the release of Java agent 4.x and EOP version 3.9.5, the legacy Contrast Java agent (3.x and earlier) can be used to analyze Java 6 and 7 web applications running on legacy technologies including:

- Jboss 4.2, 6.1, 7.1
- Weblogic 9, 10, 11

The legacy agent includes a JNDI rule in Protect which provides protection against CVE-45046.

Steps

In agent versions 3.15 and below, for certain features like Route Coverage to function you will need to set additional properties.

1. Open the YAML configuration file in an editor.
2. In addition to the pre-populated authentication keys, add the `agent.java.standalone_app_name` property.
   This property assigns the name of your application as you'd like to see it in Contrast. In this example, replace `<MyAppName>` with the name you'd like to use:

   ```
   api:
     url: https://xxx.contrastsecurity.com/Contrast
     api_key: A2xxxxxxxxxxxxxxxxxxxxxxxxxxxxG9N
     service_key: 88xxxxxxxxxxx5Z
     user_name: agent_xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx@OrgName
   agent:
     java:
       standalone_app_name: <MyAppName>
   ```

   The legacy agent can be downloaded from Maven Central Repository Search.

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 753). 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	• 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.

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 Framework Contrast tray (page 176):** 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 128)
- View supported technologies (page 126)
- View system requirements (page 127)
- Use the agent with IIS Express (page 172)
- Use application pools in IIS (page 178)

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	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. **Not supported:** • 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	

Technology	Supported versions	Notes
CLR	CLR4	
Web servers	• IIS • IIS Express	
Application frameworks	• ASP.NET MVC 3-5 • ASP.NET Web Forms • ASP.NET Web Pages • IIS-Hosted ASMX-based Web Services • IIS-Hosted Web API • IIS-Hosted WCF Services • OWIN Hosted Web API (via a Windows service or a command line application)	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). Not supported: • Analysis of .NET Framework ASP.NET Core applications (use our .NET Core agent (page 180) 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.
- Previous versions of NET 3.5 applications are supported with the Legacy .NET Framework agent (page 170).

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	• Windows 10 • Windows Server 2012, 2012 R2, 2016, 2019 • Azure Virtual Machines, Cloud Services, Mobile Services • Azure App Service	
Processor architecture	• 32-bit • 64-bit	On 64-bit systems, you can use the agent to analyze both 32-bit and 64-bit web applications.

Requirements	Recommended	Notes
CPU	At least 4	Minimum: 2
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	• .NET Framework 4.7.1 • CLR 4 (.NET 4.0 and later).	For servers on earlier versions use the Legacy .NET Framework agent (page 170).

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 174), 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 (page 170).

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:

1. Download the installer and place it on the server.
2. Run the installer.
3. 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 (page 170).

Specifically, installation varies depending on how you want to install the .NET Framework agent:

- .NET Framework Windows installer (page 128)
- Azure App Service (page 131)
- Into a container, like Docker (page 133)
- Web API-OWIN (page 135)

To auto-upgrade your agent, enable this option with the Agent Upgrade Service (page 138).

.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 Window 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 Contrast tray (page 176) application. This 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 135) (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 (page 170).

Install the .NET Framework agent using Contrast:

Install the .NET Framework agent using Contrast

1. In the Contrast web application, select **Add new** in the top right.
2. Choose **.NET Framework** in the application language dropdown, then select **IIS hosted** and select the link to **Download the agent and YAML configuration file**.
3. 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 140) 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 174).
4. You can further configure the agent using the .NET Framework YAML template (page 142).
5. 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 178) 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 specifies the install directory. Program files will be written to this directory. Defaults are dependent on OS variables.	`INSTALLFOLDER="D:\Programs\Contrast"`
`DATAFOLDER`	This 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"`
`StartTray`	When set to 0, this suppresses the start of the tray application when agent installation is completed. This is recommended when installing the agent on Windows Server Core instances. The default is 1.	`StartTray=0`
`PathToYaml`	This specifies a custom YAML configuration file. The default is the *contrast_security.yaml* file located relative to the installer's location.	`PathToYaml=c:\contrast_security.yaml`
`SERVICE_STARTUP_TYPE_MANUAL`	This must be provided when installing, upgrading and repairing the agent. If set to 1, this option sets the Contrast service startup type to Manual. The default is 0 (Automatic Delayed Start).	`SERVICE_STARTUP_TYPE_MANUAL=1`
`SUPPRESS_SERVICE_START`	This must be provided when installing, upgrading and repairing the agent. If set to 1, this option suppresses automatically starting the service. The default is 0.	`SUPPRESS_SERVICE_START=1`
`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`
`INSTALL_UPGRADE_SERVICE`	Set to 0 to not install the agent upgrade service. Default value is 1.	`INSTALL_UPGRADE_SERVICE=1`

Option	Description	Example
`UPGRADE_SERVICE_INSTALLFOLDER`	Specify the location where the installer will place 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 -
norestart StartTray=0 PathToYaml=C:\Temp\custom.yaml
```

This command installs the .NET agent in silent and unattended mode, suppresses the start of the tray application, 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

NOTE

If you don't have access to the site extension, you can manually install the .NET Framework agent with NuGet (page 135).

Steps

To install the .NET Framework agent using Azure Portal Extensions:

1. Create an application hosted on Azure App Service.
   To do this you must have an Azure account and create an ASP.NET Framework web app. Publish your application to Azure, and confirm that it works as expected without Contrast.
2. Configure settings that allow the agent to connect to Contrast by adding the following values in the **Application Settings** section of the **Configuration** blade for your application.

Key	Value
CONTRAST__API__USER_NAME	Replace with your agent username (page 32).
CONTRAST__API__SERVICE_KEY	Replace with your agent service key (page 32).
CONTRAST__API__API_KEY	Replace with your agent API key (page 32).
CONTRAST__API__URL	Defaults to *https://app.contrastsecurity.com*. Replace with another URL, if you're using a Contrast application that's hosted elsewhere.

3. In the Azure Portal, select your hosted application.
4. Select **Extensions**.
5. Select **Add**.
6. Select the **Contrast .NET Site Extension**. This is the extension for .NET Framework applications.

7. Select **OK**, and agree to the terms and conditions.
8. Wait a few seconds and confirm the site extension installed correctly.
9. Go back to the application overview and **Restart** the application.
10. Navigate to the application, and confirm the application is reporting to Contrast.

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 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.

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.S
iteExtension\ContrastAppService\ContrastProfiler-32.dll
COR_PROFILER_PATH_64=D:\home\siteextensions\Contrast.Net.Azure.S
iteExtension\ContrastAppService\ContrastProfiler-64.dll
CONTRAST_INSTALL_DIRECTORY=D:\home\siteextensions\Contrast.Net.A
zure.SiteExtension\ContrastAppService\
MicrosoftInstrumentationEngine_ConfigPath32_ContrastX86Config=D:
\home\siteextensions\Contrast.Net.Azure.SiteExtension\ContrastCi
eProfiler-32.config
MicrosoftInstrumentationEngine_ConfigPath64_ContrastX64Config=D:
\home\siteextensions\Contrast.Net.Azure.SiteExtension\ContrastCi
eProfiler-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 COR_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 COR_PROFILER* variables will be used to load the Contrast agent.

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
```

In this example, the latest .NET Framework agent is copied. 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 31) 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 33) 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 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 51) This table contains 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. **NOTE** Create application access groups in Contrast before using this variable.	`CONTRAST__APPLICATION__GROUP`
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 51). 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\ContrastProfiler-32.dll \
    COR_PROFILER_PATH_64=C:\Contrast\ContrastProfiler-64.dll
```

Additionally, the following environment variables (page 32) 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}
```

Get the API values (agent keys (page 32)) from the Contrast web interface or by downloading a YAML file (page 142) 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 131) 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-64.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 32) 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 128).
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}

Environment variable	Value
COR_PROFILER_PATH_32	*C:\Program Files\Contrast\dotnet\ContrastProfiler-32.dll*
COR_PROFILER_PATH_64	*C:\Program Files\Contrast\dotnet\ContrastProfiler-64.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\ContrastProfiler-32.dll*
COR_PROFILER_PATH_64	*C:\Program Files\Contrast\dotnet\ContrastProfiler-64.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\ContrastProfiler-64.dll"
    [string]
    $ProfilerPath64 = "C:\Program \
Files\Contrast\dotnet\ContrastProfiler-64.dll",

    # Path to the 32-bit Contrast profiler DLL.
    # Defaults to: "C:\Program \
Files\Contrast\dotnet\ContrastProfiler-32.dll"
    [string]
    $ProfilerPath32 = "C:\Program \
Files\Contrast\dotnet\ContrastProfiler-32.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:

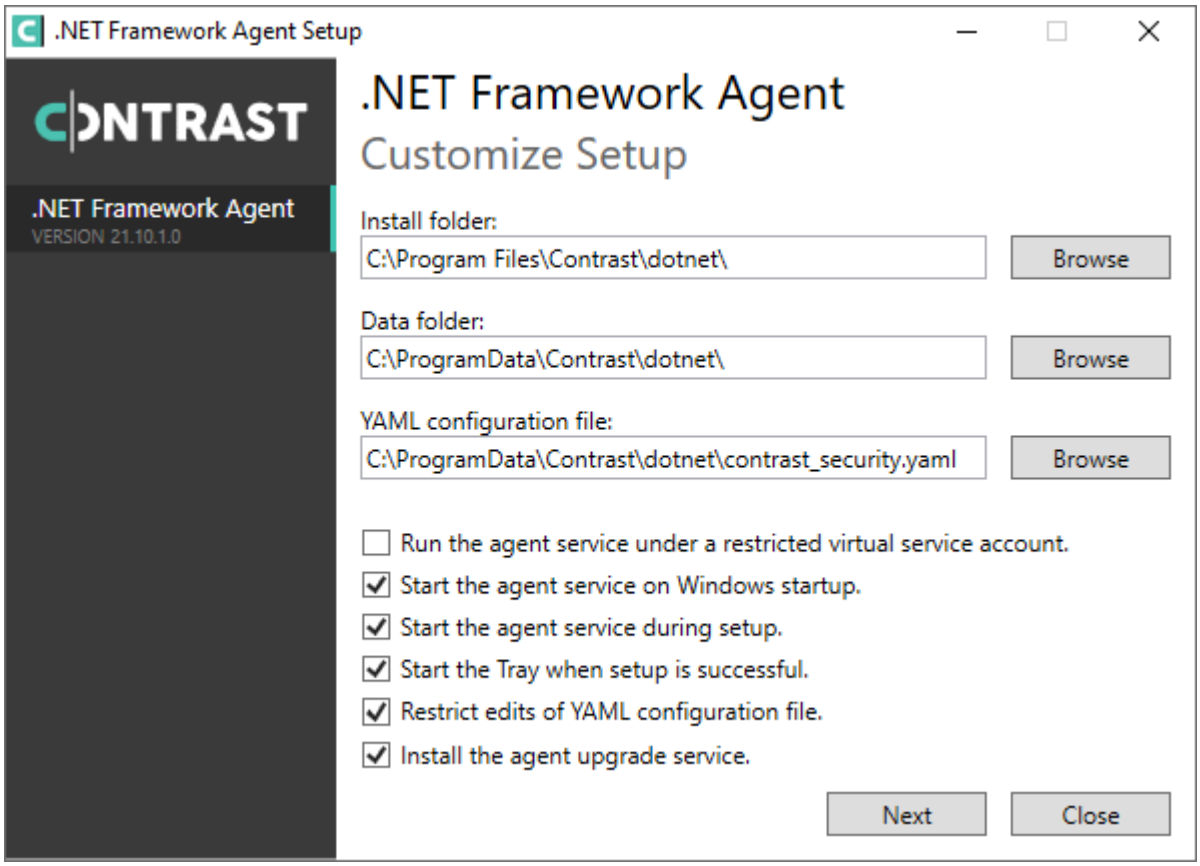
- the corresponding agent, and
- the Agent Upgrade Service.

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.
```

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

The .NET Framework has two versions, legacy and modern. The method of keeping the agent up-to-date will depend on which of the legacy/modern agent you are using.

There are three ways to update the .NET Framework agent:

- Automatically update the agent
- Download the agent from the JFrog repository (legacy agent versions only)
- Use 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 (modern)

The modern agent automatically updates through the Agent upgrade service (page 138). Contrast supports and releases new versions of the modern agent, versions 20.5.1 and later. The modern agent will not auto-update if it detects an unsupported environment (page 126).

Download the agent from the JFrog repository (legacy)

You can download the agent from the JFrog Artifactory if you need to stay on the legacy agent. This agent does not automatically update, and needs to be manually updated.

1. Download the new version.
2. Use the silent installer with command line to update the agent:

```
ContrastSetup.exe -s -norestart
```

Use the Contrast API to download the agent (modern)

1. Download the Contrast agent directly from the Contrast API. This 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 command line, after downloading the agent.
   You can find your credentials by viewing your organization and personal keys (page 436).

See also

- Agent upgrade service (page 138)

Configure the .NET Framework agent

The standard configuration (page 30) uses this order of precedence (page 31).

Configure the .NET Framework agent:

- For Azure App Service (page 141)
- In the `web.config` file (page 141)
- With the .NET Framework YAML template (page 142)

TIP
Use the Contrast agent configuration editor (page 34) 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 51).
- 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 141) 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>
     ...
```

See the .NET Framework YAML template (page 142) 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
`contrast.application.code`	19.6.3
`contrast.application.group`	19.1.3
`contrast.application.metadata`	19.1.3
`contrast.application.name`	19.1.3
`contrast.application.session_id`	20.6.6
`contrast.application.session_metadata`	20.6.6

Properties	Introduced with this .NET Framework agent version
`contrast.application.tags`	19.1.3
`contrast.application.version`	19.1.3
`contrast.assess.tags`	19.1.3
`contrast.inventory.tags`	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 33) 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

  # 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.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. 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.
    # 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.
    # 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 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

  # 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.
    # 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:

  # 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: false

  # ================================================================================
====
  # protect.rules
  # Use the following properties to set simple rule configurations.
  # ================================================================================
====
  # rules:

    # Define a list of Protect rules to disable in the agent.
    # 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.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 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

  # Override the reported server environment. Valid
  # values include `QA`, `PRODUCTION` and `DEVELOPMENT`.
  # environment: development

  # 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

  # 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.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. 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.
# 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.
# 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 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

# 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.
    # 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:

  # 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: false
```

```
  # ============================================================================
====
  # protect.rules
  # Use the following properties to set simple rule configurations.
  # ============================================================================
====
  # rules:

    # Define a list of Protect rules to disable in the agent.
    # 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.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 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

  # Override the reported server environment. Valid
  # values include `QA`, `PRODUCTION` and `DEVELOPMENT`.
  # environment: development

  # 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.

Install the .NET Framework Agent (legacy)

IMPORTANT

If your applications are running on modern technology you should use the .NET Framework agent (page 126) instead of the legacy agent.

The legacy .NET Framework agent offers the full capabilities of Contrast Assess, Protect, and SCA as of April 2020 but is no longer receiving new features or lower severity bug fixes. Contrast will continue to fix critical bugs in the legacy agent until January 10th, 2023.

You must have Editor role permission to install an agent.

The legacy Contrast .NET Framework agent can be used to analyze ASP.NET web applications running on legacy technologies including:

- Windows Server 2008
- Windows machines that cannot upgrade to .NET Framework version 4.7.1 or newer
- Applications that target the CLR2 runtime and cannot be upgraded to CLR4

See the legacy .NET Framework agent supported technologies (page 171) and system requirements (page 172) for more details.

To install the legacy .NET Framework agent:

1. Download the legacy .NET Framework agent (for example *ContrastSetup_20.4.X.zip*) where 20.4.X is the newest legacy version available.
2. Find the agent keys (page 32) and create a YAML configuration file called *contrast_security.yaml*. Copy and paste this sample and replace the `<Values>` below with your values:

```
api:
  url: <YourContrastURL>
  api_key: <YourAPIkey>
  service_key: <YourServiceKey>
  user_name: <YourUserName>
agent:
  dotnet:
    enable_chaining: <True/False>
```

3. Copy the *contrast_security.yaml* configuration file and the agent zip file to the web server.
4. Extract the downloaded zip archive (for example, *ContrastSetup_20.4.X.zip*) on the web server.
5. Move the *contrast_security.yaml* file in the same directory as *ContrastSetup.exe*.
6. Run `ContrastSetup.exe`. This installs the .NET agent.
   Once installed, the .NET 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).

Supported technologies for the .NET Framework (legacy)

IMPORTANT

The legacy .NET Framework agent offers the full capabilities of Contrast Assess, Protect, and SCA as of April 2020 but is no longer receiving new features or lower severity bug fixes. Contrast will continue to fix critical bugs in the legacy agent until January 10th, 2023.

The legacy Contrast .NET Framework agent can be used to analyze ASP.NET web applications using the following technologies:

Technology	Supported versions
Contrast version	Contrast version 3.7.3 or later
Agent versions	20.4.4 and future versions of 20.4.X
Runtime	3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8
Application framework	• ASP.NET MVC 3-5 • ASP.NET Web Forms • ASP.NET Web Pages • IIS-Hosted ASMX-based Web Services • IIS-Hosted Web API • IIS-Hosted WCF Services

Technology	Supported versions
Web servers	• IIS • IIS Express

.NET Framework (legacy) system requirements

Before installing the .NET Framework legacy 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.

Requirement	Recommended
Runtime	.NET Framework 4.5.1 present
Operating systems	• Windows 7, 8, 10 • Windows Server 2008 R2, 2012, 2012 R2, 2016, 2019
Processor architecture	The agent can be installed on both 32-bit and 64-bit systems. On 64-bit systems, you can use the agent to analyze both 32-bit and 64-bit web applications.
CPU	2
Memory	4 GB

Use the .NET Framework agent with IIS Express

The .NET agent can analyze ASP.NET applications hosted on IIS Express but it takes a little bit of work to enable instrumentation on IIS Express. The Contrast tray displays a tab for IIS Express if IIS Express is installed on the server.

The Contrast tray (page 176) initially displays a **Set environment variables** button to enable instrumentation of IIS Express-hosted applications. Selecting this button sets environment variables for the current user so that any new IIS Express process will load the .NET agent's profiler, and be instrumented and analyzed.

Once you set environment variables, the tray displays a **Remove environment variables** button that you can use to disable Contrast analysis of IIS Express-hosted applications.

Any instrumented applications currently running on IIS Express are displayed in the IIS Express tab along with a count of the number of URLs (without the `querystring`) observed.

NOTE

IIS Express process instances are commonly launched by other programs such as Visual Studio or a command window. You should restart these programs after setting these user environment variables. Any programs (such as Visual Studio) that were running before you set user environment variables will consequently launch IIS Express without the environment variables, and the IIS Express-hosted application won't be instrumented and analyzed.

Setting user environment variables also causes any .NET applications launched by the user to load the Contrast Profiler. The Contrast Profiler will safely detach from any non-IIS/non-IIS Express process. Windows treats detachment of a profiler DLL as an error message in the Windows Event Log; however, you can safely ignore these errors.

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 135) or Azure Portal Extension (page 131).

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 133). 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>
```

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\ContrastProfiler-32.dll" />
            <EnvironmentVariable Name="COR_PROFILER_PATH_64" \
Value=".\contrastsecurity\ContrastProfiler-64.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

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 140) 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
COR_PROFILER	CONTRAST_CCC_COR_PROFILER
COR_PROFILER_PATH	CONTRAST_CCC_COR_PROFILER_PATH
COR_PROFILER_PATH_32	CONTRAST_CCC_COR_PROFILER_PATH_32
COR_PROFILER_PATH_64	CONTRAST_CCC_COR_PROFILER_PATH_64

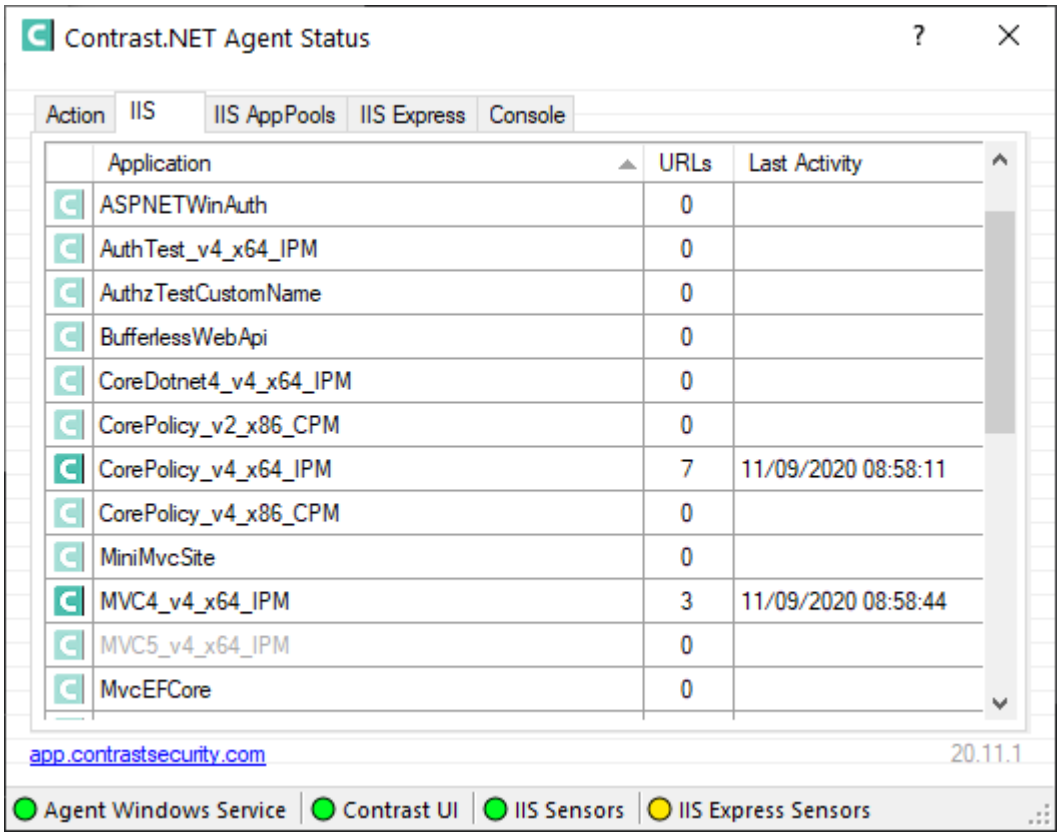
  Then follow the usual setup instructions for your environment and application.
- **AppInsights in Azure App Service:**
  As of version 20.9.3, the Contrast .NET Framework Site Extension now supports compatibility with Application Insights (using the CLR Instrumentation Engine (CIE)). There is no further 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 Framework Contrast tray

The .NET Framework Contrast tray is a Windows system tray application (*ContrastTray.exe*) that displays high-level information about the health of the agent.

NOTE

You do not have to run the .NET Framework Contrast tray to analyze applications with Contrast. It exists only to provide status information about the agent. This helps verify that the agent works as expected, especially when you initially install the agent.



The Contrast tray provides these status indicators:

- **Agent Windows Service** displays a green light when you correctly install and run the Contrast service.
- **Contrast** displays a green light when the Agent Windows Service can communicate with the Contrast application. The most common communication failure is incorrect proxy settings.
- **IIS Sensors** displays a green light when you successfully instrument an application hosted on IIS. A yellow light indicates you did instrument an application with the agent, but IIS hasn't loaded the application yet.
- **IIS Express Sensors** displays a green light for applications hosted on IIS Express when the application loads and you correctly install and run the Contrast agent. A yellow light indicates you correctly installed the Contrast agent, but IIS Express did not load the application yet. A red light indicates you have not set environment variables for IIS Express.

Select tabs in Contrast tray for more information and help:

- **Action:** Select this to see high-level user instructions for the .NET Framework agent. The instructions change based on the state of the agent. For example, if the agent can't connect to Contrast, the Action tab provides details on the error and suggestions on how to resolve the problem.
- **IIS:** Select this to see a list of all web applications running on the IIS server. The name displayed matches the alias that IIS uses to identify the application unless you specify a custom application name for Windows (page 140). The URLs column displays the number of unique URLs (not including the query string) that the agent observed for the application. The Last Activity column displays the time of the last request analyzed by the agent for that application.
- **IIS AppPools:** Select this to display a list of all application pools on the IIS server. You can see configuration details for each application pool: architecture, pipeline mode, CLR version, and identity. You can also see whether Contrast will analyze applications in this application pool. You can configure application pool filtering (page 178) with IIS for the .NET Framework agent.
- **IIS Express:** Select this to display a list of all web applications running on IIS Express.

- **Console:** Select this tab to see status and error messages that will help you troubleshoot problems with the Contrast .NET agent.

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 178) 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 *(XXXXX_Profiler_[AppDomain]XXXXX[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 33).

.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 753). 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 .NET Core later than 1.5.17	Hosted or on-premises Contrast instance	
.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	Metrics on IIS-hosted applications, including: • 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 • 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 .NET Core later than 1.7.5	Rule mode (i.e. Monitor vs. Block) for each Protect rule
.NET Framework later than 21.4.2 .NET Core later than 1.8.4	Exceptions thrown and caught within agent sensor code, including log message, exception type, exception message, and stack trace frames for `System` and `Contrast` methods.
.NET Framework later than 21.7.1 .NET Core later than 1.9.7	• Process Architecture (x86/x64)
	OS Architecture (x86/x64)
	In Azure App Service, the values of the following environment variables: • `WEBSITE_PHYSICAL_MEMORY_MB` • `WEBSITE_PLATFORM_VERSION` • `WEBSITE_SKU`
.NET Framework later than 21.9.2 .NET Core later than 2.0.1	Description of location where YAML config file was loaded from (i.e., path specified by environment variable, default location, application directory).

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.

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 182), 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 181) and these system requirements (page 182).

.NET Core supported technologies

We support the following technologies for this agent.

Technology	Supported versions	Notes
Application frameworks	• ASP.NET Core (3.1.X, 5.0.X, 6.0.X, 7.0.X) • Model-View-Controller (MVC) • Razor Pages	**Not supported:** • .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)

Runtime	• .NET Core Runtimes: 3.1.X, 5.0.X, 6.0.X , 7.0.X
	• .NET Core target framework monikers:
	  • netcoreapp3.1
	  • net5.0
	  • net6.0
	  • net7.0
		Not supported:
		• 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	• Windows Server (LTSC) (x86, x64): 2012 R2, 2016, 2019 • Windows Server (SAC) (x64): 1809, 1903 • Windows workstation (x86, x64): 7, 8/8.1, 10	On 64-bit systems, you can use the agent to analyze both 32-bit and 64-bit web applications. **Not supported:** • Windows on ARM
Server container	Kestrel, IISHttpServer	**Not supported:** Http.sys (formerly called WebListener)
Hosting container	Self-hosted, IIS, IIS Express	
.NET Core for Linux		
Linux operating systems	• Ubuntu: 16.04 and later • Debian: 9 and later • openSUSE: 15 and later • Alpine: 3.7 and later • CentOS: 7 and later • Red Hat Enterprise Linux: 7, 8	All systems require x64 architectures. 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.

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 Core system requirements

Before installing the .NET Core agent confirm you can meet the following requirements:

- You have administrative access to a server, and the server is supported by Contrast (page 181).
- There is a deployed application to be analyzed, and the web application technology is supported (page 181) by Contrast.
- The web server has network connectivity with Contrast.
- The server meets the minimum requirements (stated below).

Requirement	Recommended	Minimum	Notes
.NET Core versions	3.1.X, 5.0.X, 6.0.X, 7.0.X	2.1, 3.0, 3.1	
CPU	at least 4	2	
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.
Processor architecture	• 32-bit processors • 64-bit processors		The .NET Core agent does **not** support systems with ARM-based processors (for example, Apple M1 processors). This restriction includes the use of the agent within containers on these systems.

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 182) (if you are using self-hosted web application running on Windows, Linux or Docker)
- .NET Core agent for IIS installer (page 189) (if you are using IIS)
- Azure App Service (page 187)
- NuGet (page 186)

To auto-upgrade your agent, enable this option with the Agent Upgrade Service (page 138).

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.

Before you begin, check the system requirements (page 182) and supported technologies (page 181) to be sure installation will work and ensure best performance.

NOTE
Installing within containers can be complex, and these steps might not work for your situation. Read more about installing with Docker.

1. Select **Add new** at the top right of the Contrast web interface. Choose **.NET Core** and select the link to download the .NET Core 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 31) 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**

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>`

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 198)	Configures the agent	Yes; set the environment variable CONTRAST_CONFIG_PATH	Read
<UnzippedDirectoryRoot>	The root "installation" directory; stores the agent binaries	No	Read

Path	Usage	Customizable	Permissions
• **Windows:** *%ProgramData%\Contrast\dotnet-core\logs* • **Linux:** */var/tmp/contrast/dotnet-core/logs*	Directory for Contrast agent logs. If missing, the directory will be created	Yes; set the environment variable CONTRAST_CORECLR_LOGS_DIRECTORY	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 197) 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 195), 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 184)
- Bash (Linux) (page 185)
- Powershell or Powershell Core (Windows) (page 185)
- Launch profile (dotnet.exe) (page 185)

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)

```
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
```

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 187) or if you prefer to include the .NET Core agent as a dependency.

To manually install the .NET Core agent using NuGet:

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:

   ```
   CORECLR_ENABLE_PROFILING: 1
   CORECLR_PROFILER: {8B2CE134-0948-48CA-A4B2-80DDAD9F5791}
   CORECLR_PROFILER_PATH: <application directory>/contrast/runtimes/linux-
   x64/native/ContrastProfiler.so
   ```
3. Set the basic configuration either with the YAML configuration file (page 198) or with environment variables (page 197). 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.

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.

Install the .NET Core agent with Azure App Service

Before you begin, check the system requirements (page 182) and supported technologies (page 181) to be sure installation will work and ensure best performance.

To complete an express installation of the .NET Core agent using Azure Portal Extensions:

1. Create an Azure account, if you don't have one already.
2. Create a .NET Core 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.)

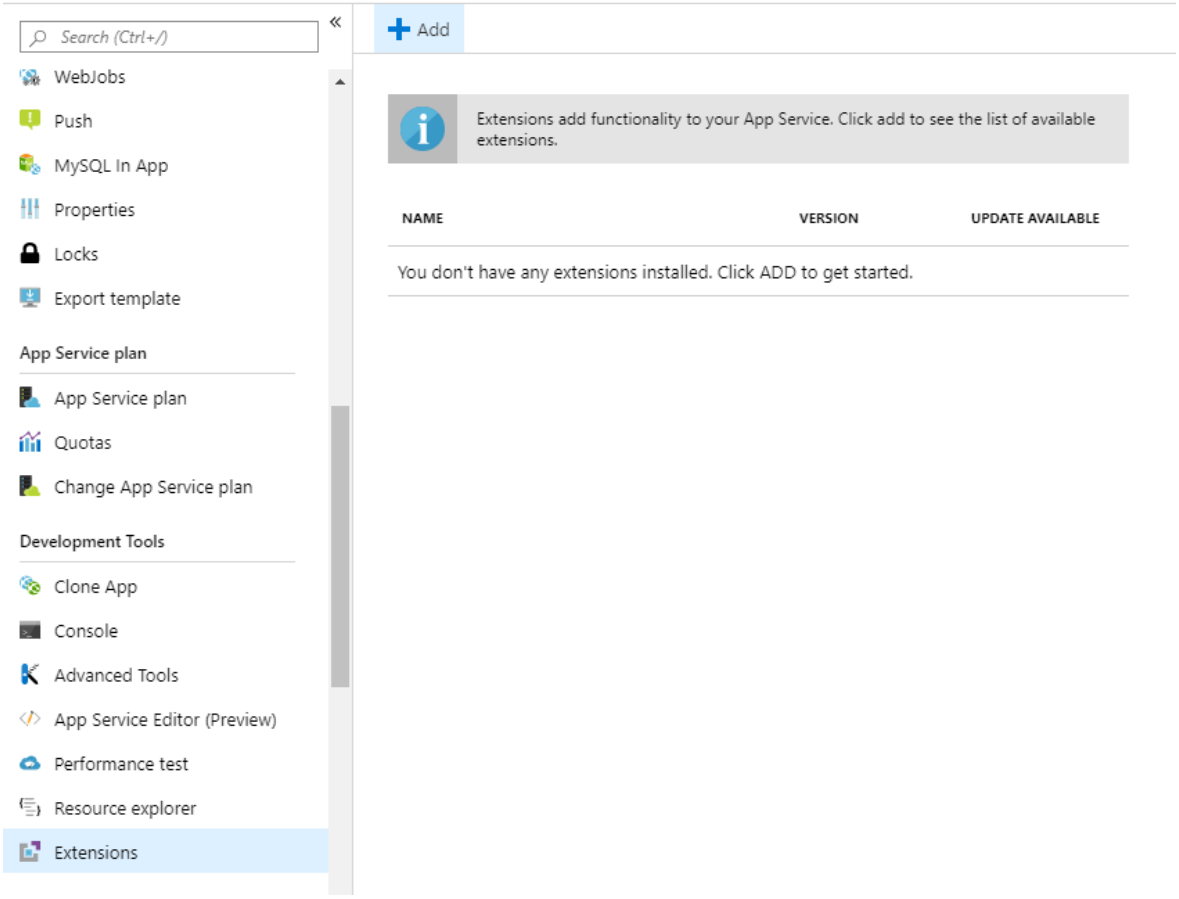
NOTE

If you do not have access to the site extension, you can install the .NET Core agent manually with NuGet (page 186).

5. In the Azure Portal, select your hosted application.
6. Select **Configuration** under **Settings** to configure settings that allow the agent to connect to Contrast.
7. Select **New application setting** and add the following values for your application:

Key	Value
CONTRAST__API__USER_NAME	Replace with your agent username (page 32).
CONTRAST__API__SERVICE_KEY	Replace with your agent service key (page 32).
CONTRAST__API__API_KEY	Replace with your agent API key (page 32).
CONTRAST__API__URL	Defaults to *https://app.contrastsecurity.com*. Replace with another URL, if you're using a Contrast application that's hosted elsewhere.

8. Select **Extensions**.



9. Select **Add**.
10. Select the **Contrast .NET Core Site Extension for Azure App Service**. This is the extension for .NET Core applications.
11. Select **OK**, and agree to the terms and conditions.
12. Wait a few seconds and confirm the site extension installed correctly.
13. Go back to the application overview and **Restart** the application.
14. Navigate to the application, and confirm the application is reporting to Contrast.

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.

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.NetCo
re.Azure.SiteExtension\ContrastNetCoreAppService\
MicrosoftInstrumentationEngine_ConfigPath32_ContrastCoreX86Confi
g=D:\Home\siteextensions\Contrast.NetCore.Azure.SiteExtension\Co
ntrastCieCoreClrProfiler-32.config
MicrosoftInstrumentationEngine_ConfigPath64_ContrastCoreX64Confi
g=D:\Home\siteextensions\Contrast.NetCore.Azure.SiteExtension\Co
ntrastCieCoreClrProfiler-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.

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 182) and supported technologies (page 181) 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 198) to set the authentication keys (page 32) 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`	Specify the location where the installer will place agent files.	`INSTALLFOLDER=C:Program Files\Contrast\dotnet-core`
`DATAFOLDER`	Specify the default location for agent log and configuration files.	`DATAFOLDER=C:\ProgramData\Contrast\dotnet-core`
`SUPPRESS_RESTARTING_IIS`	When set the installer will not restart IIS. Note that the agent will not be loaded by applications until IIS is restarted.	`SUPPRESS_RESTARTING_IIS=0`
`UPGRADE_SERVICE_INSTALLFOLDER`	Specify the location where the installer will place 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.

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 31) 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 33), 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 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 51). 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. **NOTE** Application access groups have to be created first in Contrast.	`CONTRAST__APPLICATION__GROUP`
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 182). The `CORECLR_` variables load the agent and the `CONTRAST_` variables are for agent authentication to the server.

Using the Dockerfile example from before:

```
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}
```

Additionally, the following environment variables (page 32) 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 32)) 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.

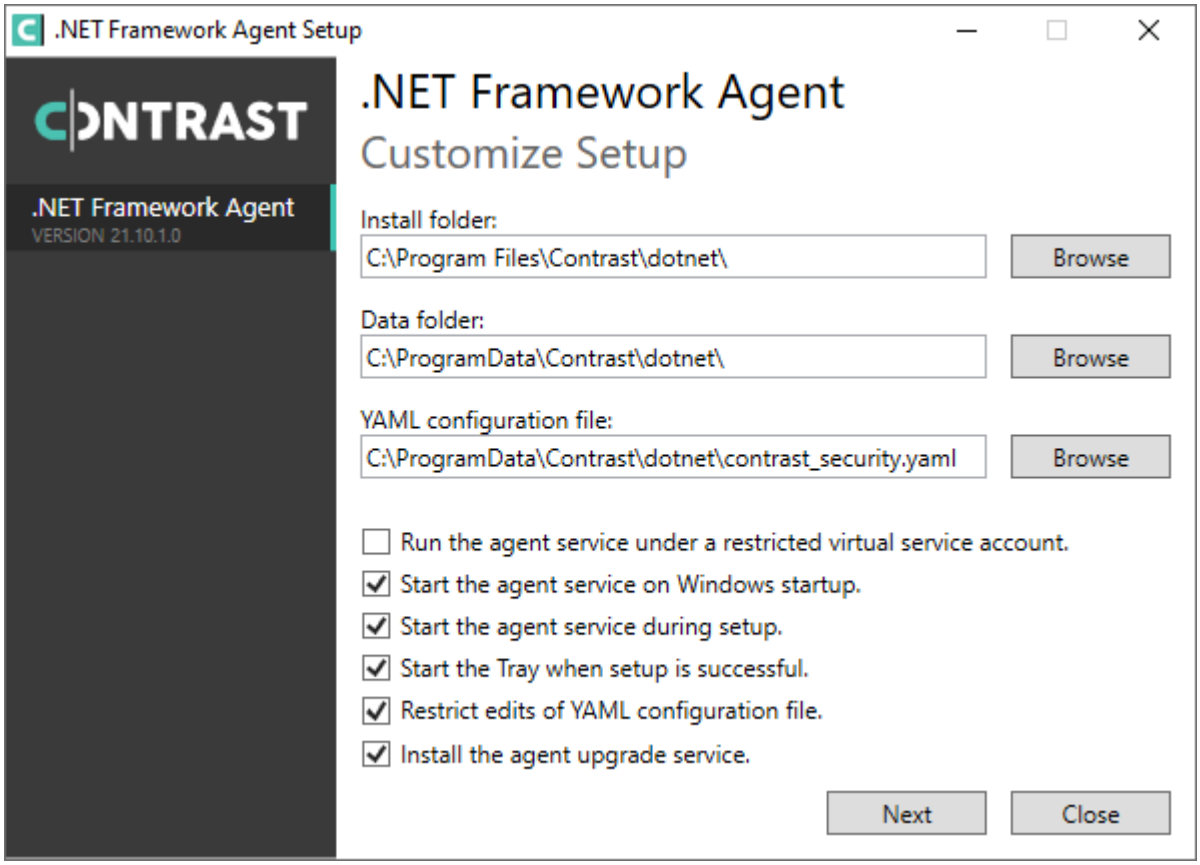
NOTE

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.

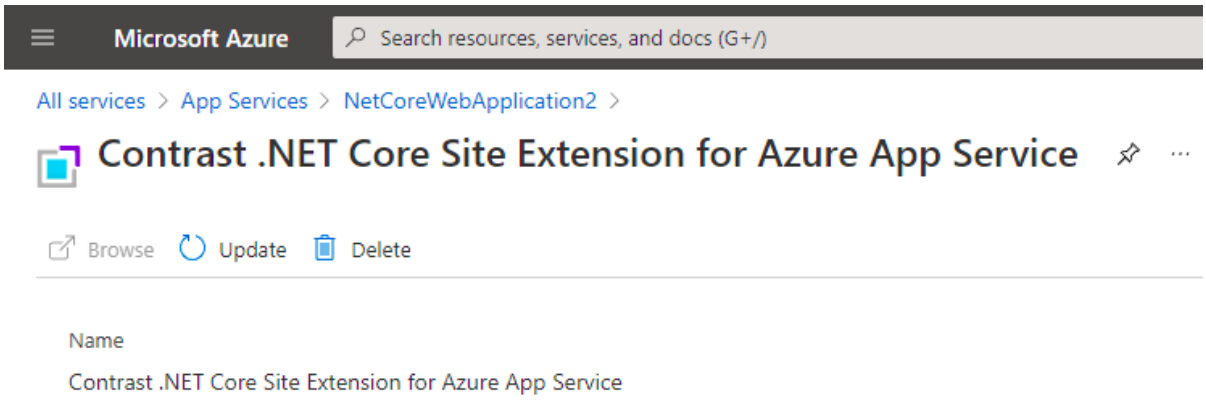
```
enable: true #  Set to `true` for the agent to automatically upgrade to \
newer versions.
checks: Startup # Set the frequency with which the agent checks for \
updates. Valid values are `daily` for every 24 hours and on startup, or \
`startup` for *only* when service starts up.
timeout_ms: 60000 # 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.
```

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 138).
- **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 30) for all agents uses this order of precedence (page 31).

Depending on your situation, you can configure the .NET Core agent with:

- Azure App Service (page 197)
- Environment variables (page 197)
- A YAML configuration file (page 198)
- Integrations (page 553)

TIP

Use the Contrast agent configuration editor (page 34) to create or upload a YAML configuration file, validate YAML and get setting recommendations.

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 197).
  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 198):** 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 187)

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 198) 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 636).)
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 198) 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 33).)

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

  # 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.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. 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 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 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.
    # 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:

  # 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: false

  # ================================================================================
====
  # protect.rules
  # Use the following properties to set simple rule configurations.
  # ================================================================================
====
  # rules:

    # Define a list of Protect rules to disable in the agent.
    # 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.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 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

  # Override the reported server environment. Valid
  # values include `QA`, `PRODUCTION` and `DEVELOPMENT`.
  # environment: development

  # 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

  # 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.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. 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 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 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.
    # 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:

  # 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: false
```

```
  # ================================================================================
====
  # protect.rules
  # Use the following properties to set simple rule configurations.
  # ================================================================================
====
  # rules:

    # Define a list of Protect rules to disable in the agent.
    # 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.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 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

  # Override the reported server environment. Valid
  # values include `QA`, `PRODUCTION` and `DEVELOPMENT`.
  # environment: development

  # 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
```

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	Nuget Package	Windows
AppDynanics	21.8.1	6.0	Installed	Windows
AppDynamics	21.8.1	6.0	Nuget Package	Windows
Dynatrace One Agent	1.253.245	6.0	Installed	Windows, Ubuntu
New Relic	8.23.107	6.0	NuGet Package	Windows
New Relic	8.23.1076.	6.0	NuGet Package	Ubuntu

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 50) will automatically setup chaining and is the preferred route to add agents to existing Kubernete workloads.

Manual

Chaining may need to be 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 182).

.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 753). 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 .NET Core later than 1.5.17	Hosted or on-premises Contrast instance	
.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	Metrics on IIS-hosted applications, including: • 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 • 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 .NET Core later than 1.7.5	Rule mode (i.e. Monitor vs. Block) for each Protect rule
.NET Framework later than 21.4.2 .NET Core later than 1.8.4	Exceptions thrown and caught within agent sensor code, including log message, exception type, exception message, and stack trace frames for `System` and `Contrast` methods.
.NET Framework later than 21.7.1 .NET Core later than 1.9.7	• Process Architecture (x86/x64)
	OS Architecture (x86/x64)
	In Azure App Service, the values of the following environment variables: • `WEBSITE_PHYSICAL_MEMORY_MB` • `WEBSITE_PLATFORM_VERSION` • `WEBSITE_SKU`
.NET Framework later than 21.9.2 .NET Core later than 2.0.1	Description of location where YAML config file was loaded from (i.e., path specified by environment variable, default location, application directory).

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	• Windows application: supported with Azure .NET Framework Site Extension	• Docker Linux image • Linux application
2.x	.NET Core 3.1 (unless pinned to ~2.0, which makes it .NET Core 2.1)	• Windows Application: supported on Azure	• Windows Application: Not locally supported because the host uses .NET Core 2.1 • Linux application
3.x	.NET Core 3.1, .NET 5	• Windows Application: Supported locally and in Azure with the .NET Core Site Extension • Docker Linux Image: Supported locally and in Azure	• Linux application
4.x	.NET 6	• Windows Application: Supported locally and in Azure with the .NET Core Site Extension • Docker Linux Image: Supported locally and in Azure	• Linux application

NOTE
For all versions, running the Azure Functions application in isolated mode is not supported by the .NET agent.

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, 2 (not pinned to ~2.0), 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) or CI/CD, the site extension automatically sets the required application settings. In these cases, you can still specify the application settings manually, if desired.

Specifying the application settings manually is required if you use the Azure Functions portal editor to deploy the Azure Function. 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.SiteE
xtension\ContrastNetCoreAppService-<version>\runtimes\win-
x86\native\ContrastProfiler.dll
CORECLR_PROFILER_PATH_64=C:\home\SiteExtensions\Contrast.NetCore.Azure.SiteE
xtension\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\ContrastNetCoreA
ppService-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 prior to execution.

The Contrast Node.js agent follows semantic versioning (major.minor.patch). The agent works best with these supported technologies (page 230) and these system requirements (page 231).

The Node.js agent rewrites the application code prior to startup using the Babel compiler. After start up the agent patches the required functions for the supported frameworks and modules (page 230).

Once you install the Node.js agent (page 232), 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

The Contrast service (page 430) 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 431) with environment variables (page 51) or a YAML configuration file (page 33). 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 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	
• JavaScript ECMAScript 5 • JavaScript ECMAScript 6 • ECMAScript modules (ESM) • TypeScript	**Notes** Contrast supports even numbered Node.js versions in "active LTS" or "maintenance" status. The Node.js LTS versions support these features for JavaScript ECMAScript5 and 6. The Contrast Node.js agent provides limited support for working with user apps that use ESM. TypeScript is only supported if the agent is configured to point to the compiled entry point for your application.
Node.js Long-Term Support (LTS)	
All versions in Active and Maintenance LTS status, currently: • 12* and 14 • 16 (only for agent version 4.5.0 and later) • 18 (only for agent version 4.25.0 and later)	**Notes** You should always use Node.js LTS versions that are active or in maintenance status. *Although Contrast continues to support version 12 LTS, it reached EOL at the end of April 2022. This EOL version presents a security risk. Also, note that version 5.X.X of the agent requires a minimum Node.js version of 14.15.0. The Node.js agent doesn't guarantee support for Node.js features classified as Experimental (Stability: 1). It also doesn't instrument the native `net` module. It only provides functionality for `HTTP(S)` application servers built using the supported application frameworks in this table. `HTTP/2` is supported for the Node.js agent when using the Node.js core `HTTP/2` or `spdy` library. For customer applications using `HTTP/2` with Contrast Node.js agent, you must configure the agent to use `assess.enable_lazy_tracking: false`. Node.js version status is shown in Node.js Long-Term Support Release Schedule.
Node package manager (npm)	

`npm` versions: • >=6.13.7 • >=7.11.0 • >= 8.5.5	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.
Application frameworks	
• 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)	
• 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	
• Joi 17 and later • Validator 13 and later • Class-validator 0.13.0 and later	
Templating engines	
• Handlebars 4 • Pug 3 • EJS 2.6.2, 3.0.1 • Mustache 4.x and later	
Other technologies	
• 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.

Operating systems

- 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

- PM2: 4.5.0 and later and 5.1.0 and later
  - 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 `@contrast/agent` module. For example:

```
node -r /Users/michael/Dev/my-app/node_modules/@contrast/agent app.js
```

Containers

- Distroless containers
  - The Contrast Node.js agent requires access to `npm` and `/bin/sh` to function correctly. The agent's functionality will be degraded and may not report findings or libraries when installed in an application that runs on a distroless container image that does not include those programs/packages.

CPU

- AMD 64, x86_64 and compatible
- Apple M1/M2 ARM64 (limited support)

Install the Node.js agent

There are several ways to install the agent depending on your situation, but generally, this is the process:

1. Get the Node.js agent from npm.
2. Set authentication keys (page 32) .
3. Add a command to the `package.json` file to enable your application to run with the agent.
4. Run your application with the agent.
5. Use your application as you normally would and verify that Contrast sees the application.

To avoid errors, follow these specific instructions depending on how your application is deployed:

- Install manually (page 232)
- Install in a container (page 233)
- Install with IBM Cloud (page 236)

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
   ```

   NOTE
   If you don't want to install optional dependencies included with the Node.js agent, append the `--no-optional` CLI flag to your install command or add `optional=false` to your NPMRC file.

2. Set authentication keys (page 32) with environment variables (page 51). Make sure your Node.js application has access to the environment variables at runtime.
   Alternatively, set the configuration with YAML configuration (page 33) using this template (page 240). 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 -r @contrast/agent <app-main>.js",
       "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 239) 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

Before you begin

This topic provides general guidance for installing the Contrast 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 --no-optional
```

- **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 into your existing Dockerfile or into a new Dockerfile that uses your application's image as a base image.

```
npm install @contrast/agent --no-optional
```

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 239)). Configuration for the Node.js agent follows this order of precedence (page 31).

1. Create a YAML configuration file.
   Your `contrast_security.yaml` file should reside in your application's directory so it is copied to the container file system.
   In the YAML file:
   - Replace <YourURL>, <YourUserName>, <YourAPIKey> and <YourServiceKey> with your values. Find your agent keys here (page 32).

   A typical YAML file for a container installation might look like this:

```
api:
  url: <URLtoContrast>
  user_name: <YourUserName>
  api_key: <YourAPIKey>
  service_key: <YourServiceKey>
agent:
  service:
    grpc: true
assess:
  enable_lazy_tracking: false
```

2. Copy the YAML file into the base image using this command (in this example, `/app/contrast_security.yaml` is the base directory for your application in the image).

```
COPY WORKSPACE/contrast_security.yaml /app/contrast_security.yaml
```

3. 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 239) commonly used to set application-specific values.

Run and verify

1. If you want to use the Node.js agent rewriter CLI (available in version 4.X and later), add a `RUN` instruction and provide the command to invoke the `contrast-transpile` executable. Then provide your application's entry point.

```
RUN npx -p @contrast/agent --no contrast-transpile index.js
```

   It is important to use the `-p @contrast/agent --no` options to ensure the **npx** command is the one from Contrast Security and not from a nefarious person attempting a supply chain attack.
   `-p` is the shorthand for `--package`, which tells **npx** to only use the command in the `@contrast/agent package`.
   `--no` is the new option name for the deprecated `--no-install` option that tells **npx** to not attempt an install from npm if the command binary is not found.
   The expectation is that the Contrast agent has already been correctly installed along with the **npx** binaries before attempting to run the **npx** command.

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”, “-r”, “@contrast/agent”, “app”]
```

3. When the agent starts, it will try to connect to Contrast with authentication keys (page 32) in the YAML configuration file.

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:

```
@contrast/agent 2.16.8------------------------------------
2020-07-20T19:05:14.407Z INFO contrast-service: BUILD {"progname": \
"Contrast Service", "version": "2.8.1", "buildTime": ""}
2020-07-20T19:05:14.407Z INFO Building timer for orphan request cleanup \
{"progname": "Contrast Service", "cleanupMs": 5000}
2020-07-20T19:05:14.408Z INFO Building timer for orphan app cleanup \
{"progname": "Contrast Service", "time": 5000}
2020-07-20T19:05:14.450Z INFO Creating New Application Server \
{"progname": "Contrast Service", "uuid": "96299b72-f867-4354-
b9c9-1eb23511cb8a", "serverName": "bc1bd6e5cd3a", "clientId": "1", \
"pid": 1}
2020-07-20T19:05:14.450Z WARN Failed to initialize secure client, \
falling back to insecure client {"progname": "Contrast Service"}
2020-07-20T19:05:15.473Z INFO setting new server features for \
context{"progname": "Contrast Service", "uuid": "96299b72-f867-4354-
b9c9-1eb23511cb8a", "serverName": "bc1bd6e5cd3a"}
2020-07-20T19:05:15.474Z ERROR Error setting up CEF syslog {"progname": \
"Contrast Service", "err": "open /juice-shop/security.log: permission \
denied"}
2020-07-20T19:05:15.475Z INFO starting event scanner {"progname": \
"Contrast Service", "report": {}}
2020-07-20T19:05:15.486Z INFO Creating new application {"progname": \
"Contrast Service", "uuid": "96299b72-f867-4354-
b9c9-1eb23511cb8a", "serverName": "bc1bd6e5cd3a", "appName": "juiceshop-
guide", "language": "Node", "clientId": "1", "pid": 1}
2020-07-20T19:05:15.486Z INFO AppCreate: creating and initializing new \
application {"progname": "Contrast Service", "uuid": "96299b72-f867-4354-
b9c9-1eb23511cb8a", "server_name": "bc1bd6e5cd3a", "app_name": "juiceshop
-guide", "app_lang": "Node", "client_id": "1", "pid": 1}
2020-07-20T19:05:15.921Z INFO setting new application settings \
```

```
{"progname": "Contrast Service", "uuid": "96299b72-f867-4354-
b9c9-1eb23511cb8a", "serverName": "bc1bd6e5cd3a", "appName": "juiceshop-
guide", "language": "Node"}
2020-07-20T19:05:15.922Z INFO Setting session id on app context: \
{"progname": "Contrast Service", "uuid": "96299b72-f867-4354-
b9c9-1eb23511cb8a", "clientid": "1", "appname": "juiceshop-
guide", "applang": "Node", "apppath": "/juice-shop/
package.json", "sessionid": "cd0b271e66974162bf5fcca8b32e37b1"}
Entering main at /juice-shop/appinfo: All dependencies in ./
package.json are satisfied (OK)...
```

See also

Contrast Support Portal Node.js agent with Kubernetes

Contrast Support Portal AWS Fargate and Contrast agents

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 239) using a YAML configuration file to set the authentication keys (page 32) 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. Create a folder named *contrast* in your application's root directory. and move the *node-contrast-.tgz* and the *contrast_security.yaml* files into the *contrast* folder.
5. Add this command to the `"scripts":` section of your application's *package.json* file:

   ```
   "ibmcloud-with-contrast": "npm install @contrast/agent && node -
   r @contrast/agent index.js -c /home/vcap/app/contrast/
   contrast_security.yaml",
   ```
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": "npm install @contrast/agent && node -
   r @contrast/agent index.js -c /home/vcap/app/contrast/
   ```

```
contrast_security.yaml",
"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 contrast
```

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.

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 -r @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 --no-optional
     ```
3. After installing with either method, you will see output like this:

   ```
   $ npm install @contrast/agent

   > grpc@1.24.4 install /Users/<aUserName>/Documents/test-apps/juice-shop/
   node_modules/grpc> node-pre-gyp install --fallback-to-build --
   library=static_library

   node-pre-gyp WARN Using request for node-pre-gyp https download[grpc] \
   Success: "/Users/<aUserName>/Documents/test-apps/juice-shop/node_modules/
   grpc/src/node/extension_binary/node-v72-darwin-x64-unknown/
   grpc_node.node" is installed via remotenpm WARN jest-config@26.6.1 \
   requires a peer of ts-node@>=9.0.0 but none is installed. You must \
   install peer dependencies yourself.npm WARN jsdom@16.4.0 requires a \
   peer of canvas@^2.5.0 but none is installed. You must install peer \
   dependencies yourself.npm WARN ws@7.3.1 requires a peer of \
   bufferutil@^4.0.1 but none is installed. You must install peer \
   dependencies yourself.npm WARN ws@7.3.1 requires a peer of utf-8-
   ```

```
validate@^5.0.2 but none is installed. You must install peer \
dependencies yourself.

+ @contrast/agent@3.4.0added 19 packages from 43 contributors, updated \
5 packages and audited 1995 packages in 14.904sfound 19 vulnerabilities \
(5 low, 7 moderate, 4 high, 3 critical)
  run `npm audit fix` to fix them, or `npm audit` for details
```

4. To check whether the installation/update succeeded, run the following command and look for this output:

```
$ npm list | grep contrast
 @contrast/agent@3.4.0
  @contrast/distringuish-prebuilt@2.0.0
  @contrast/escodegen@1.16.0
  @contrast/esprima@4.1.1
  @contrast/estraverse@5.1.0
  @contrast/flat@4.2.0
  @contrast/fn-inspect@2.3.0
  @contrast/heapdump@1.0.0
  @contrast/protobuf-api@2.2.3
  @contrast/require-hook@1.1.2
  @contrast/synchronous-source-maps@1.1.0
```

See also

- Node.js supported technologies (page 230)
- Install Node.js (page 232)

Configure the Node.js agent

The standard configuration (page 30) for all agents uses this order of precedence (page 31).

There are several ways to configure the Node.js agent, but generally, you should:

- Use a YAML configuration file to set configuration values that are common for all applications in an organization or container (for example, to redirect logging or proxy configuration). This template (page 240) shows all valid configuration options for the Node.js agent. Learn more about YAML configuration (page 33) in general.
- Use environment variables (page 239) for agent authentication keys and application-specific configuration values. Learn more about environment variables in general (page 51).

TIP

Use the Contrast agent configuration editor (page 34) 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 240), but here are some common examples as environment variables:

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.
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__AGENT__LOGGER__APPEND	When set to `false`, creates a new log file on startup instead of appending and rolling daily. Default is `true`.
CONTRAST__AGENT__LOGGER__LEVEL	Logging level: `FATAL, ERROR, WARN, INFO, DEBUG or TRACE`. Default is `ERROR`.
CONTRAST__AGENT__LOGGER__PATH	Where Contrast will put its debug log. Default is *node-contrast.log*.
CONTRAST__AGENT__LOGGER__STDOUT	When set to `false`, suppresses output to `stdout`. Default is `true`.

NOTE

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 33).)

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

  # 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. 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:

  # Set to limit the length of Error stack traces to a specified number.
  # stack_trace_limit: 10

  # ================================================================================
====
  # 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 `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.
    # 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. 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.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 location of the application's `package.json` file.
    # app_root: NEEDS_TO_BE_SET

    # ============================================================================
====
    # agent.node.rewrite_cache
    # Use the following properties to set up rewrite caching in the agent.
    # ============================================================================
====
    # rewrite_cache:

      # Set to `true` to enable rewrite caching.
      # enable: false

      # Set the location of the rewrite cache source.
      # path: NEEDS_TO_BE_SET

# ============================================================================
====
# inventory
# Use the properties in this section to override the inventory features.
# ============================================================================
====
# inventory:
```

```
  # 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

  # 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

  # 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

# ============================================================================
====
# 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: 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 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

  # Override the reported server environment. Valid
  # values include `QA`, `PRODUCTION` and `DEVELOPMENT`.
  # environment: development

  # 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

  # 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. 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:

  # Set to limit the length of Error stack traces to a specified number.
  # stack_trace_limit: 10

  # ================================================================================
====
```

```
  # 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 `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.
    # 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. 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.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 location of the application's `package.json` file.
    # app_root: NEEDS_TO_BE_SET
```

```
    # ============================================================================
====
    # agent.node.rewrite_cache
    # Use the following properties to set up rewrite caching in the agent.
    # ============================================================================
====
    # rewrite_cache:

      # Set to `true` to enable rewrite caching.
      # enable: false

      # Set the location of the rewrite cache source.
      # path: NEEDS_TO_BE_SET

# ============================================================================
====
# inventory
# Use the properties in this section to override the inventory features.
# ============================================================================
====
# inventory:

  # 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

  # 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

  # 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

# ================================================================================
====
# 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: 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 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

  # Override the reported server environment. Valid
  # values include `QA`, `PRODUCTION` and `DEVELOPMENT`.
  # environment: development

  # 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
```

Node.js agent rewriter CLI

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.

Use the rewriter

1. In the Node.js agent's configuration file (`contrast_security.yaml`):
   - Enable rewrite caching and specify the path of the cache location.
   - Specify the logging level.

NOTE

You can override the YAML configuration settings by invoking the rewriter with the CLI arguments or environment variables (page 239). For example, to prevent any logging to stdout or to disk you could use these overrides:

```
npx -p @contrast/agent --no contrast-transpile index.js --
agent.logger.stdout false --agent.logger.path /dev/null;
```

It is important to use the `-p @contrast/agent --no` options to ensure the **npx** command is the one from Contrast Security and not from a nefarious person attempting a supply chain attack.

- `-p` is the shorthand for `--package`, which tells **npx** to only use the command in the `@contrast/agent package`.
- `--no` is the new option name for the deprecated `--no-install` option that tells **npx** to not attempt an install from npm if the command binary is not found.
  The expectation is that the Contrast agent has already been correctly installed along with the **npx** binaries before attempting to run the **npx** command.

- You must also explicitly enable Protect or Assess.

For example:

```
agent:
  logger:
    level: info
    path: ./node-contrast
  node:
    rewrite_cache:
      enable: true
      path: ./rewrite_cache
assess:
  enable: true
protect:
  enable: false
```

2. Invoke the executable and provide it with your application's entry point (for example, `index.js`). For example:

```
npx -p @contrast/agent --no contrast-transpile index.js --
agent.logger.level trace;
...
trace: 2021-07-12T18:31:15.128Z 2934603 contrast:rewrite -
 successfully rewrote code for /path/to/app/index.js
trace: 2021-07-12T18:31:15.186Z 2934603 contrast:rewrite -
 successfully rewrote code for /path/to/app/node_modules/koa/lib/
application.js
trace: 2021-07-12T18:31:15.251Z 2934603 contrast:rewrite -
 successfully rewrote code for /path/to/app/node_modules/koa-router/lib/
router.js
trace: 2021-07-12T18:31:15.314Z 2934603 contrast:rewrite -
 successfully rewrote code for /path/to/app/node_modules/@koa/router/lib/
router.js
...
info: 2021-07-12T18:31:41.608Z 2946030 contrast:cli-rewriter -
 rewriting complete [26.625s]
```

> **NOTE**
> The Node.js agent rewriter CLI currently only supports pre-compiling applications to run with Assess enabled. The agent transpiles code differently depending on whether Protect or Assess are enabled. Protect requires fewer source transformations than Assess and does not cause the same startup delays.

3. Once rewriting completes, start your application with Contrast as you usually would. (for example, `node -r @contrast/agent index.js`).

Use the Node.js agent with ESM

The Contrast Node.js agent provides limited support for using ECMAScript modules (ESM) in Node.js server-side applications. ESM is the official standard format to package JavaScript client-side code and now Node.js provides support for ESM.

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 wiith 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.

> **NOTE**
> The Contrast Node.js agent does not support ESM Conditional Exports.
>
> ECMAScript instrumentation is experimental and Contrast requires at least Node.js version 14.15.0 for support.

When instrumenting an application that uses ESM (or a combination of both ESM and CJS), start the application like this:

```
  Usage: node --experimental-loader @contrast/agent/esm.mjs
app-main.mjs [agent arguments] -- [app arguments]
```

Transpilers, compilers, source maps and the Node.js agent

The Node.js agent supports applications written in languages that compile to JavaScript, such 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 original source.

Some languages, like TypeScript, require you to precompile your code before runtime. In these cases, the Node.js agent must point to the compiled entrypoint for your application.

To do this, set up the Node.js agent using the -r option:

```
scripts: {
    "test": "...",
    "start": "...",
    "contrast": "node -r @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 rewriter 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 753). 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

Agent versions	Data
	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 currently supports Assess and SCA only.

As a next step, you can:

- Install the PHP agent (page 263)
- View PHP agent system requirements (page 263)
- View PHP agent supported technologies (page 263)

PHP agent supported technologies

We support the following technologies for this agent.

Technology	Supported versions	Notes
Language version	• 7.4, 8.0, 8.1 (NTS)	The agent depends on the `mbstring` and `curl` extensions.
Operating systems	• Debian • RHEL/CentOS	
Application frameworks	• Laravel • Drupal 8, 9	
Processor architecture	• AMD64	
Servers	• Apache	Other servers using `mod_php` or `php-fpm` may work but are not currently supported.
Package manager	• Composer	Supported by SCA analysis.

PHP agent system requirements

Before installing the PHP agent, your system must meet the following requirements:

Requirement	Version	Notes
Runtime system	• 64-bit Linux	

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 264)
- Install PHP with RPM (page 265)

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:

```
sudo apt-get update && sudo apt-get install contrast-php-agent
```

> **NOTE**
> Once PHP is configured to use the extension, it will be used whenever the interpreter is executed. This step should be delayed until immediately before the application itself is run. This will prevent any unexpected behavior when running artisan or other PHP commands.

3. Locate the PHP configuration file, called `php.ini` and 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.
4. Edit your PHP configuration file:

```
echo "extension=/usr/local/lib/contrast/php/contrast.so" >> `php-
config --ini-path`/php.ini
```

5. Configure the PHP agent (page 266) using the PHP YAML template (page 266) or environment variables.
6. Start your application in the normal way.
7. Exercise and test your application.
8. Verify that the PHP agent is running by checking the Contrast UI and/or looking for PHP agent log output (depending on configuration).

Notes

- It is possible to use the agent with the PHP CLI by adding a flag to the command line:

```
php -d extension=/usr/local/lib/contrast/php/contrast.so
```

- Library analysis and route discovery is 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 when any other third-party extensions (including xdebug, APMs, etc.) is undefined.
- 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-$releasever repo
   baseurl=https://pkg.contrastsecurity.com/rpm-public/centos-$releasever/
   gpgcheck=0
   enabled=1
   EOF
   ```

2. Once complete, use this command to install the agent and service:

   ```
   sudo yum install contrast-php-agent
   ```

   NOTE

   Once PHP is configured to use the extension, it will be used whenever the interpreter is executed. This step should be delayed until immediately before the application itself is run. This will prevent any unexpected behavior when running artisan or other PHP commands.

3. Locate the PHP configuration file, called `php.ini` and 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.
4. Edit your PHP configuration file:

   ```
   echo "extension=/usr/local/lib/contrast/php/contrast.so" >> `php-
   config --ini-path`/php.ini
   ```

5. Configure the PHP agent (page 266) using the PHP YAML template (page 266) or environment variables.
6. Start your application in the normal way.
7. Exercise and test your application.
8. Verify that the PHP agent is running by checking the Contrast UI and/or looking for PHP agent log output (depending on configuration).

Notes

- It is possible to use the agent with the PHP CLI by adding a flag to the command line:

```
php -d extension=/usr/local/lib/contrast/php/contrast.so
```

- Library analysis and route discovery is 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 when any other third-party extensions (including xdebug, APMs, etc.) is undefined.
- 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.

Configure the PHP agent

The standard configuration (page 30) for all agents uses this order of precedence (page 31).

You may use a YAML configuration file or environment variables to configure the agent when running your application:

- You can create your own YAML configuration file or use this YAML template (page 266) that contains all valid properties for the PHP agent,
- or you can use environment variables (page 51) to configure your build.

TIP

Use the Contrast agent configuration editor (page 34) 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 33).)

Place your YAML file in the working directory of your application or 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

  # ================================================================================
====
  # 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. 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.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

    # ********************** REQUIRED **********************
```

```
    # Timeout (in milliseconds) that the agent-init
    # should wait for the contrast-service
    timeout_ms: 30000

    # Set to `true` to enable direct communication with Teamserver.
    # bypass: false

    # ================================================================================
    # 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.go
  # The following properties apply to any Go agent-wide configurations.
  # ================================================================================
  # go:

    # Enable opt-in Go agent features.
    # preview: NEEDS_TO_BE_SET

    # Enable Go agent self-profiling features.
    # profile: 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

# ================================================================================
====
# 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.
    # 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:

  # 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: false
```

```
# ============================================================================
====
# application
# Use the properties in this section for
# the application(s) hosting this agent.
# ============================================================================
====
# application:

  # 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.
  #
  # 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

  # Override the reported server environment. Valid
  # values include `QA`, `PRODUCTION` and `DEVELOPMENT`.
  # environment: development

  # 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

  # ============================================================================
====
  # 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. 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.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

    # ********************** REQUIRED **********************
    # Timeout (in milliseconds) that the agent-init
    # should wait for the contrast-service
    timeout_ms: 30000

    # Set to `true` to enable direct communication with Teamserver.
```

```
    # bypass: false

    # ================================================================================
    # 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.go
  # The following properties apply to any Go agent-wide configurations.
  # ================================================================================
  # go:

    # Enable opt-in Go agent features.
    # preview: NEEDS_TO_BE_SET

    # Enable Go agent self-profiling features.
    # profile: 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

# ================================================================================
====
# 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.
    # 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:

  # 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: false

# ================================================================================
====
# application
# Use the properties in this section for
```

```
# the application(s) hosting this agent.
# ================================================================================
====
# application:

  # 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.
  #
  # 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

  # Override the reported server environment. Valid
  # values include `QA`, `PRODUCTION` and `DEVELOPMENT`.
  # environment: development

  # 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
```

PHP YAML template

Use this template to configure the PHP agent using a YAML configuration file. (Learn more about YAML configuration (page 33).)

Place your YAML file in the working directory of your application or 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

  # ================================================================================
====
  # 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. 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.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

    # ********************** REQUIRED **********************
    # Timeout (in milliseconds) that the agent-init
    # should wait for the contrast-service
    timeout_ms: 30000
```

```
    # Set to `true` to enable direct communication with Teamserver.
    # bypass: false

    # ================================================================================
====
    # 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.go
  # The following properties apply to any Go agent-wide configurations.
  # ================================================================================
====
  # go:

    # Enable opt-in Go agent features.
    # preview: NEEDS_TO_BE_SET

    # Enable Go agent self-profiling features.
    # profile: 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

# ================================================================================
====
# 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.
    # 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:

  # 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: false

# ================================================================================
====
```

```
# application
# Use the properties in this section for
# the application(s) hosting this agent.
# ================================================================================
====
# application:

  # 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.
  #
  # 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

  # Override the reported server environment. Valid
  # values include `QA`, `PRODUCTION` and `DEVELOPMENT`.
  # environment: development

  # 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

  # ============================================================================
====
  # 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. 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.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

    # ********************** REQUIRED **********************
    # Timeout (in milliseconds) that the agent-init
    # should wait for the contrast-service
    timeout_ms: 30000

    # Set to `true` to enable direct communication with Teamserver.
    # bypass: false

    # ================================================================================
```

```
====
    # 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.go
  # The following properties apply to any Go agent-wide configurations.
  # ============================================================================
====
  # go:

    # Enable opt-in Go agent features.
    # preview: NEEDS_TO_BE_SET

    # Enable Go agent self-profiling features.
    # profile: 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
```

```
# ================================================================================
====
# 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.
    # 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:

  # 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: false

# ================================================================================
====
# application
# Use the properties in this section for
# the application(s) hosting this agent.
# ================================================================================
====
```

```
# application:

  # 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.
  #
  # 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

  # Override the reported server environment. Valid
  # values include `QA`, `PRODUCTION` and `DEVELOPMENT`.
  # environment: development

  # 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
```

Python agent

The Python agent is a WSGI- and framework-specific middleware that's compatible with the most popular web application frameworks. The agent's goal is to be fully WSGI compatible, along with other web frameworks, and to provide applications built on WSGI or those frameworks 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 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. The service 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 Python agent supports Assess, Protect, and SCA.

As a next step, you can:

- Install the agent (page 296)
- View supported technologies for the Python agent (page 295)

System requirements for the Python agent

WARNING
The Python agent does not currently support running on Macs with the M1 chip.

Before installing the Python 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 PyPI or the agent manually installed.
- The server meets the minimum requirements shown in this table.

Requirement	Versions	Notes
Operating system	• 64-bit OSX • 64-bit Linux	Starting with version 2.3.0 of the agent, the package installation step requires the compilation of C extensions. This process is automatic, but it requires that certain software is installed in the target environment: • Required: `gcc`, `make`, `automake` and `autoconf`. The package names may be different on different platforms. Installing your platform's version of `build-essential` or installing system headers may be necessary. If running an agent on Alpine OS, `libtool` is required.
Python packages	• protobuf: 3.12 and later • psutil: 5.7 and later • pip: 6 and later	

Supported technologies for the Python agent

We support the following technologies for this agent.

Technology	Supported versions	Notes
Language version	• 3.10.X: First supported agent was 5.2.0 • 3.9.X: First supported agent was 4.2.0 • 3.8.X: First supported agent was 2.8.0 • 3.7.X: First supported agent was 1.5.0	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. **Not supported:** • 3.6.X, 3.5X and 2.7.X: Last supported agent was 4.14.3

Technology	Supported versions	Notes
Application frameworks	• Aiohttp 3.7 and later • Bottle: 0.11, 0.12 • Django: 1.11, 2.X , 3.X and 4.X • Django Rest Framework: 3.12 and later • Falcon (ASGI): 3.0, 3.1 • Falcon: (WSGI) 2.X and 3.X • FastAPI: 0.68* - 0.79* • Flask: 1.X, 2.X • Pyramid: 1.10 and 2.X • Quart: 0.15 and later	The Python agent is meant to be WSGI compatible. It may be compatible to other WSGI applications as long as the guidelines are followed. The Agent is also ASGI compatible for the FastAPI framework only. * 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.
Processor architecture	The agent is tested on x86_64	It may work on other architectures but it isn't officially supported.
Web servers	• Gunicorn 0.16.1 –20.1.X • uWSGI 2.0.14 - 2.0.X • Uvicorn: 0.14* - 0.17*	
Databases	• Mongo (`pymongo`) • MySQL (`PyMySQL` and `mysql-connector`) • PostgreSQL (`psycopg2`) • SQLite3 (`sqlite3` and `pysqlite2`)	
Object-relational mapping databases (ORM)	• Flask-SQLAlchemy • SQLAlchemy	

Install the Python agent

The Python agent is installed as a standard Python package, and uses the Contrast service to communicate results.

By default, the service is started automatically when your application starts. It's also possible to configure the agent to communicate with a standalone Contrast service (page 430) that runs independently.

Install the agent with PyPi (page 296) or update the Python agent (page 297).

WARNING

The Python agent does not currently support running on Macs with the M1 chip.

Install the Python agent with PyPI

To install the Python agent with PyPI:

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`.

2. Configure middleware. (page 321)

3. Configure the agent. (page 298)
4. Verify that `autoconf` is installed on the system where you will run the agent.
5. Start and exercise your application as normal.
6. 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 295)
- Install Python (page 296)

Configure the Python agent

The standard installation (page 30) for all agents uses this order of precedence (page 31).

Additionally, you must configure middleware (page 321) and configure the Python agent with a YAML configuration file.

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 31) 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 34) 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 33).)

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

  # Set the path to which the agent should store the
  # currently used configuration from the Contrast UI.
  # last_config_path: ./tmp/config

  # ============================================================================
====
  # 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

  # ================================================================================
====
  # agent.security_logger
  # Define the following properties to set security
  # logging values. 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.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

    # 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.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.
    # 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:

  # 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: false

  # ================================================================================
====
  # protect.rules
  # Use the following properties to set simple rule configurations.
  # ================================================================================
====
  # rules:

    # Define a list of Protect rules to disable in the agent.
    # 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: true

      # 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: true

      # 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: true

      # 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: true

    # ================================================================================
====
    # 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 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

  # Override the reported server environment. Valid
  # values include `QA`, `PRODUCTION` and `DEVELOPMENT`.
  # environment: development
```

```
  # 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

  # Set the path to which the agent should store the
  # currently used configuration from the Contrast UI.
  # last_config_path: ./tmp/config

  # ============================================================================
====
  # 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

  # ================================================================================
  # agent.security_logger
  # Define the following properties to set security
  # logging values. 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.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

    # 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.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.
    # 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:

  # 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: false

  # ================================================================================
====
  # protect.rules
  # Use the following properties to set simple rule configurations.
  # ================================================================================
====
  # rules:

    # Define a list of Protect rules to disable in the agent.
    # 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: true
```

```
      # 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: true

      # 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: true

      # 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: true

    # ================================================================================
    # 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 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

  # Override the reported server environment. Valid
  # values include `QA`, `PRODUCTION` and `DEVELOPMENT`.
  # environment: development

  # 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
```

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 298), environment variables (page 51), or both.
3. Run the command `contrast-validate-config`. This script validates your current configuration and tests your ability to communicate with Contrast.

See also

- Configure the Python agent (page 298)

Configure middleware

Middleware is a software component that's 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 is 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 a 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 295) 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.PyramidMi
ddleware')
```

   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, given a WSGI app 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.

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 33).)

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

  # Set the path to which the agent should store the
  # currently used configuration from the Contrast UI.
  # last_config_path: ./tmp/config

  # ============================================================================
====
  # 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

  # ================================================================================
  # agent.security_logger
  # Define the following properties to set security
  # logging values. 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.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

    # 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.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.
    # 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:

  # 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: false

  # ================================================================================
====
  # protect.rules
  # Use the following properties to set simple rule configurations.
  # ================================================================================
====
```

```
  # rules:

    # Define a list of Protect rules to disable in the agent.
    # 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: true

      # 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: true

      # 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: true

      # 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: true

    # ================================================================================
```

```
====
    # 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 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

  # Override the reported server environment. Valid
  # values include `QA`, `PRODUCTION` and `DEVELOPMENT`.
  # environment: development

  # 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

  # Set the path to which the agent should store the
  # currently used configuration from the Contrast UI.
  # last_config_path: ./tmp/config

  # ================================================================================
====
  # 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

  # ================================================================================
====
  # agent.security_logger
  # Define the following properties to set security
  # logging values. 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.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

    # 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.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.
    # 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:

  # 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: false

  # ================================================================================
====
  # protect.rules
  # Use the following properties to set simple rule configurations.
  # ================================================================================
====
  # rules:

    # Define a list of Protect rules to disable in the agent.
    # 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: true

      # 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: true

      # 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: true

      # 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: true

    # ================================================================================
====
    # 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 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

  # Override the reported server environment. Valid
  # values include `QA`, `PRODUCTION` and `DEVELOPMENT`.
  # environment: development

  # 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
```

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 753). 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.

Ruby agent

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 355)
- View Ruby agent supported technologies (page 354)

System requirements for the Ruby agent

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	• 64-bit Linux (recommended) • 64-bit OSX • 64-bit Alpine	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: • `gcc`, `make`, `automake` and `autoconf` (Package names may vary. You may install your platform's version of `build-essential`.) • system headers The agent runs in the Ruby application layer with some C dependencies so it may need to be installed with the `--platform ruby` flag to allow for compilation in either *g libc* or *musl libc* based systems.
Ruby gems	• ougai: ~>1.8 (1.8 and later, earlier than 2) • parser: ~>2.6 (2.6 and later, earlier than 3) • protobuf: ~>3.10 (3.10 and later, earlier than 4 • rack: ~>2.0 (2.0 and later, earlier than 3)	

Supported technologies for the Ruby agent

We support the following technologies for this agent.

Technology	Supported versions	Notes
Language version	• 3.1.X: First supported agent was 6.0.0 • 3.0.X: First supported agent was 4.6.0 • 2.7.X: First supported agent was 3.8.0 See the Ruby Maintenance Branches schedule for specific release dates.	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. **Not supported:** • 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

Technology	Supported versions	Notes
Application frameworks	• Rails 5.X - 7.X • Grape 1.5.X • Sinatra 3.X	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. **Not supported:** • Rails 3.X: Last supported agent was 3.11.0 • Rails 4.X: Last supported agent was 3.11.0
Databases	• MongoDB • Mysql2 • PG • SQLite3	Contrast doesn't support old or deprecated versions of third-party modules.
Testing environments	We test on a matrix of our supported Operating Systems, Application Frameworks, and Web Servers and also run the Ruby Mspec Suite.	When changes are made, Contrast runs a battery of automated tests to ensure that it detects findings in supported technologies across all supported versions of Ruby. In addition to its own testing, Contrast also runs the Ruby Spec Suite against an environment with the agent enabled.
Web servers	• Passenger 5.37 and 6.X • Puma 3.7 - 5.X • Thin 1.7.2 - 1.8.X • Unicorn 5.0.X - 6.X	

Install the Ruby agent

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 355) or update the Ruby agent (page 356).

Install the Ruby agent with RubyGems as a gem source

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 357) (Rails, Sinatra, or Grape).
4. Configure the agent (page 357).
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

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 354)
- Install Ruby (page 355)

Configure the Ruby agent

The configuration (page 30) for all agents uses this order of precedence (page 31). Configure the agent using a YAML configuration file.

TIP

Use the Contrast agent configuration editor (page 34) to create or upload a YAML configuration file, validate YAML, and get setting recommendations.

Follow these guidelines to configure your:

- Frameworks (page 378)
  - Grape (page 378)
  - Rails (page 378)
  - Sinatra (page 378)
- Web servers (page 379)

- Passenger (page 379)
- Puma (page 382)
- Thin (page 385)
- Unicorn (page 386)

See also

- Ruby supported technologies (page 354)

Ruby YAML template

Use this template to configure the Ruby agent using a YAML configuration file. (Learn more about YAML configuration (page 33).)

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

  # Set the path to which the agent should store the
```

```
  # currently used configuration from the Contrast UI.
  # last_config_path: ./tmp/config

  # ================================================================================
====
  # 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. 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 temproary 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.
    # 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:

  # 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: false

  # ================================================================================
====
  # protect.rules
  # Use the following properties to set simple rule configurations.
  # ================================================================================
====
  # rules:

    # Define a list of Protect rules to disable in the agent.
    # 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 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

  # Override the reported server environment. Valid
  # values include `QA`, `PRODUCTION` and `DEVELOPMENT`.
  # environment: development

  # 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

  # Set the path to which the agent should store the
  # currently used configuration from the Contrast UI.
  # last_config_path: ./tmp/config

  # ================================================================================
====
  # 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. 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 temproary 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.
    # 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:

  # 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: false

  # ================================================================================
====
  # protect.rules
  # Use the following properties to set simple rule configurations.
  # ================================================================================
====
  # rules:

    # Define a list of Protect rules to disable in the agent.
    # 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 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

  # Override the reported server environment. Valid
  # values include `QA`, `PRODUCTION` and `DEVELOPMENT`.
  # environment: development

  # 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 Grape

If you're 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 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're 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
```

See also

- Configure the Ruby agent (page 357)
- Ruby supported technologies (page 354)

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 379)
- Puma (page 382)
- Thin (page 385)
- Unicorn (page 386)

See also

- Configure the Ruby agent (page 357)
- Ruby supported technologies (page 354)

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 severs: *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 don 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 357)
- Ruby supported technologies (page 354)

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 357)
- Ruby supported technologies (page 354)

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 357)
- Ruby supported technologies (page 354)

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 `unicorf.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

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
`worker_processes (nr)`	Sets the current number of `worker_processes` to `nr`. Each worker process will serve exactly one client at a time. You can increment or decrement this value at runtime by sending signals `SIGTTIN` or `SIGTTOU` respectively to the master process without reloading the rest of your Unicorn configuration. You can read more about signals on Unicorn's site.
`Unicorn::Configurator#after_fork`	Sets `after_fork` hook to a given block
`Unicorn::Configurator#before_fork`	Sets `before_fork` hook to a given block.
`Unicorn::Configurator#preload_app(bool) ObjectEnabling`	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
timeout 30	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
delay integer	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 357)

- Ruby supported technologies (page 354)

Ruby YAML template

Use this template to configure the Ruby agent using a YAML configuration file. (Learn more about YAML configuration (page 33).)

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

  # Set the path to which the agent should store the
  # currently used configuration from the Contrast UI.
  # last_config_path: ./tmp/config

  # ================================================================================
====
  # 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. 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 temproary 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.
```

```
    # 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:

  # 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: false

  # ============================================================================
====
  # protect.rules
  # Use the following properties to set simple rule configurations.
  # ============================================================================
====
  # rules:

    # Define a list of Protect rules to disable in the agent.
    # 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 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

  # Override the reported server environment. Valid
  # values include `QA`, `PRODUCTION` and `DEVELOPMENT`.
  # environment: development

  # 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
```

```
  # Set the path to which the agent should store the
  # currently used configuration from the Contrast UI.
  # last_config_path: ./tmp/config

  # ================================================================================
====
  # 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. 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 temproary 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.
    # 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:

  # 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: false

  # ================================================================================
====
  # protect.rules
  # Use the following properties to set simple rule configurations.
  # ================================================================================
====
  # rules:

    # Define a list of Protect rules to disable in the agent.
    # 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 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

  # Override the reported server environment. Valid
  # values include `QA`, `PRODUCTION` and `DEVELOPMENT`.
  # environment: development

  # 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 753). 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
	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.

NOTE
As a source code rewriter, installing the Go agent in an application requires access to the application build environment.

The Go agent currently supports Assess and SCA only.

As a next step, you can:

- Install the Go agent (page 410)
- View Go agent system requirements (page 410)
- Go supported technologies (page 410)

Supported technologies for the Go agent

We support the following technologies for this agent.

Technology	Supported versions	Notes
Language version	• 1.18.X • 1.19.x	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 `go.mod` file. **Not supported:** • 1.17.X: Last supported agent was 3.6.0 • 1.16.X: Last supported agent was 3.6.0 • 1.15.X: Last supported agent was 1.12.0
Operating systems	• Linux • Darwin	
Application frameworks	• Go standard library • Gin 1.X • Chi 5.X	Other frameworks may be compatible at runtime with reduced functionality for certain features.
Processor architecture	AMD64	The Go agent rewriter and the generated runtime may work on other architectures but is not officially supported.
Database support	Go standard library	Some third-party database libraries may be compatible but are not supported.
Dependency management system	Go mod	The agent only supports modules as a dependency system: Using Go modules. An application can be initialized with modules by running `go mod init`.
HTTP request routers	`julienschmidt/httprouter`	

Go agent system requirements

Before installing the Go agent, your system must meet the following requirements:

Requirement	Version	Notes
Build system	• 64-bit OSX • 64-bit Linux	**Required:** Go build tools version 1.16 or higher. The AMD64 architectures of these operating systems are supported, not the ARM versions.
Runtime system	• 64-bit OSX • 64-bit Linux	The AMD64 architectures of these operating systems are supported, not the ARM versions.

Install the Go agent

The executable must be built using `contrast-go` to instrument the application with the agent. A basic installation of the Go agent consists of two parts: producing an executable and running the executable. An installation overview looks like this:

1. Download the following and place in the proper path:
   - Go agent
   - `contrast_security.yaml`
2. Set up appropriate permissions.
3. Build the Go application, replacing `go` with `contrast-go` to get a final application artifact which contains the Go agent.
4. Run the executable.
5. Exercise and test your application.
6. Verify that Contrast sees your application.

TIP

To see a list of available flags with command line arguments for `contrast-go`, type `contrast-go -h`.

For specific installation instructions, select one of the following options:

- Install with contrast-go-installer (page 411)
- Install Go from RPM (.rpm) (page 414)
- Install Go from Debian (.deb) (page 414)
- Install from direct download (page 415)

NOTE

Before 3.0.0, the Go agent required the Contrast service. Follow the install the Go agent (legacy) (page 412) instructions.

Install the Go agent with contrast-go-installer

The `contrast-go-installer` simplifies the installation of the Go agent.

NOTE

The `contrast-go-installer` is an open source project. You can review the code and contribute to its development here.

Steps

1. Run the `contrast-go-installer` with this command:

```
go run github.com/contrast-security-oss/contrast-go-
installer@latest latest
```

2. Build your application with this command:

```
contrast-go build -o output-name-of-application
```

3. Configure the Go agent (page 415)using the Go YAML template (page 416) or environment variables.
4. Run your application using the executable you generated in step 2.
5. Exercise and test your application.
6. In the Contrast web interface, verify that the application is in the Applications list.

Install the Go agent (legacy)

IMPORTANT

You can configure the agent to report directly to Contrast by installing the Go agent without the Contrast service. (page 410)

Using the Go agent involves two artifacts. The executable must be built using `contrast-go` to instrument the application with the agent. The application requires `contrast-service` to communicate with the Contrast dashboard at runtime. A basic installation of the Go agent consists of two parts: producing an executable and running the executable. An installation overview looks like this:

1. Download the following and place in the proper path:
   - Go agent
   - Contrast service (page 430)
   - `contrast_security.yaml`
     Ensure your yaml or environment variables are set to `bypass:false`.

     ```
     agent:
       service:
         bypass: false
     ```
2. Set up appropriate permissions.
3. Build the Go application, replacing `go` with `contrast-go` to get a final application artifact which contains the Go agent.
4. Run the executable.
5. Exercise and test your application.
6. Verify that Contrast sees your application.

TIP

To see a list of available flags with command line arguments for `contrast-go`, type `contrast-go -h`.

For specific installation instructions, select one of the following options:

- Install Go from RPM (.rpm) (page 414)
- Install Go from Debian (.deb) (page 414)
- Install from direct download (page 415)

Install the Go agent using a container

The Go agent has two main components: `contrast-go` (a build tool) and the agent itself (a package injected into your app at build time by `contrast-go`).

You can download `contrast-go` with a curl command, or from a repository or package manager. You will also need to configure the agent to connect with Contrast. You can use a YAML configuration file or environment variables. You can download the YAML from Contrast, or create your own from a template.

When installing the Contrast Go agent in a container, you just need to follow these steps in your container image file.

TIP

If you would like to explore a sample application using the Go agent in a *separate* container, see the Go Test Bench open source project.

Before you begin

This topic provides general guidance for installing the Contrast Go 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.

Build your application with `contrast-go`

Follow the instructions to install the Go agent.

1. Download the Go agent. Either use the curl below, or install with Debian (page 414).

   ```
   RUN curl -L https://pkg.contrastsecurity.com/go-agent-release/latest/
   linux-amd64/contrast-go > contrast-go
   ```

2. Ensure the correct permissions are set

   ```
   RUN chmod u+x contrast-go
   ```

3. Replace your normal `go build` command with `contrast-go build`. This will give you an executable with Contrast embedded in it.

   ```
   RUN ./contrast-go build -o instrumented-app
   ```

   NOTE

   `-o instrumented-app` names it `instrumented-app` which is used in the below steps. You can change the name or omit the -o entirely and a default name is generated based on the package.

Run your instrumented application

- Run your application.

  ```
  CMD ./instrumented-app
  ```

See also

- Kubernetes and Contrast
- AWS Fargate and Contrast agents

Install Go agent with Debian

To install the Go agent:

1. Download the executable files 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:

```
sudo apt-get update && sudo apt-get install 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 415) using the Go YAML template (page 416) 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.

Install Go agent with Red Hat Package Manager (RPM)

To install the Go agent:

1. Download the executable files 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-$releasever repo
baseurl=https://pkg.contrastsecurity.com/rpm-public/centos-$releasever/
gpgcheck=0
enabled=1
EOF
```

2. Once complete, use this command to install the agent:

```
sudo yum install 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 415) using the Go YAML template (page 416) or environment variables.
6. Run your application using the executable with the output from above.
7. Exercise and test your application.
8. Verify that Contrast sees your application.

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 415) using the Go YAML template (page 416) 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

The standard configuration (page 30) for all agents uses this order of precedence (page 31), which also details other ways to configure the agent, such as other valid configuration file locations.

There are different ways to configure the Go agent:

- A YAML configuration file (learn more) (page 33) is required to run your application. It is not needed to run the Go agent. You can create your own YAML configuration file or use this template (page 416) that contains all valid properties for the Go agent.
  Place the file in the directory your app will run from "." and then find the agent keys (page 32) and set these values:

```
api:
  url:
  api_key:
  service_key:
  user_name:
```

- You can use environment variables (page 51) to configure your build.
- Command line configuration is also available.

TIP
Use the Contrast agent configuration editor (page 34) to create or upload a YAML configuration file, validate YAML and get setting recommendations.

Go YAML template

Use this template to configure the Go agent using a YAML configuration file. (Learn more about YAML configuration (page 33).)

```
# ================================================================================
====
# 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

  # ================================================================================
====
  # 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. 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.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

    # ********************** REQUIRED **********************
    # Timeout (in milliseconds) that the agent-init
    # should wait for the contrast-service
    timeout_ms: 30000

    # Set to `true` to enable direct communication with Teamserver.
    # bypass: false

    # ================================================================================
====
    # 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.go
  # The following properties apply to any Go agent-wide configurations.
  # ============================================================================
====
  # go:

    # Enable opt-in Go agent features.
    # preview: NEEDS_TO_BE_SET

    # Enable Go agent self-profiling features.
    # profile: 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

# ============================================================================
====
# 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.
    # 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:

  # 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: false

# ================================================================================
====
# application
# Use the properties in this section for
# the application(s) hosting this agent.
# ================================================================================
====
# application:

  # 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.
  #
  # 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

  # Override the reported server environment. Valid
  # values include `QA`, `PRODUCTION` and `DEVELOPMENT`.
  # environment: development

  # 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

  # ============================================================================
====
  # 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. 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.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

    # ********************** REQUIRED **********************
    # Timeout (in milliseconds) that the agent-init
    # should wait for the contrast-service
    timeout_ms: 30000

    # Set to `true` to enable direct communication with Teamserver.
    # bypass: false

    # ============================================================================
====
    # 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.go
  # The following properties apply to any Go agent-wide configurations.
  # ================================================================================
====
  # go:

    # Enable opt-in Go agent features.
    # preview: NEEDS_TO_BE_SET

    # Enable Go agent self-profiling features.
    # profile: 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

# ================================================================================
====
# 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.
    # 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:

  # 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: false

# ================================================================================
====
# application
# Use the properties in this section for
# the application(s) hosting this agent.
# ================================================================================
====
# application:

  # 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.
  #
  # 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

# Override the reported server environment. Valid
# values include `QA`, `PRODUCTION` and `DEVELOPMENT`.
# environment: development

# 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

The Contrast service is a stand-alone executable that enables 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 431).
- **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 431)

TIP

To remove the `contrast-service` package, run `apt-get remove contrast-service` or `yum remove contrast-service`.

Configure the Contrast service

Unlike the service executable packaged with the Python agent, the Contrast service isn't preconfigured with connection parameters. Instead, 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 431).
- **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 431)

TIP

To remove the `contrast-service package`, run `apt-get remove contrast-service` or `yum remove contrast-service`.

Configure the Contrast service

Unlike the service executable packaged with the Python agent, the Contrast service isn't preconfigured with connection parameters. Instead, 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`.

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 437) under user settings. (page 435))

With Editor permissions you can instrument an application (page 30) and start viewing results in Contrast. You can also interact with the basic components of Contrast (all visible in the header of the web interface):

- Applications (page 438)
  View a searchable list of an organization's applications. License, merge, tag, archive and restore applications.
- Servers (page 483)
  View a searchable list of an organization's servers. Designate server environment, enable Assess and Protect, settings, tagging and deleting.
- Libraries (page 488)
  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 524)
  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 539)
  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 547)
  Collect data and export as a CSV or PDF to share it outside of Contrast.
- Integrations (page 553)
  Use Contrast in conjunction with other tools like bugtrackers, build tools, application servers, Security Incident Event Management (SIEM), notifications and chat.
- Contrast CLI (page 511)
  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 668), organization (page 628) or RulesAdmin (page 600) permissions, an Editor can:

- Instrument an application (page 30)
- Send notifications
- Customize scoring (page 646)

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 435)
- Set up two-step authentication (page 436)
- Edit your profile (page 436)
- Find API keys (page 436)
- Manage notifications (page 437)
- View your permissions (page 437)

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 640), 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 436), 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 639). 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 435) don't have the option to change their password.

Set up two-step authentication

If your administrator has enabled two-step authentication at an organization (page 639) or system (page 713) 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 > Two-step verification**.
2. Use the toggle to enable two-step authentication.
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 two-step 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 436) under **Your keys**.
5. Select **Save**.

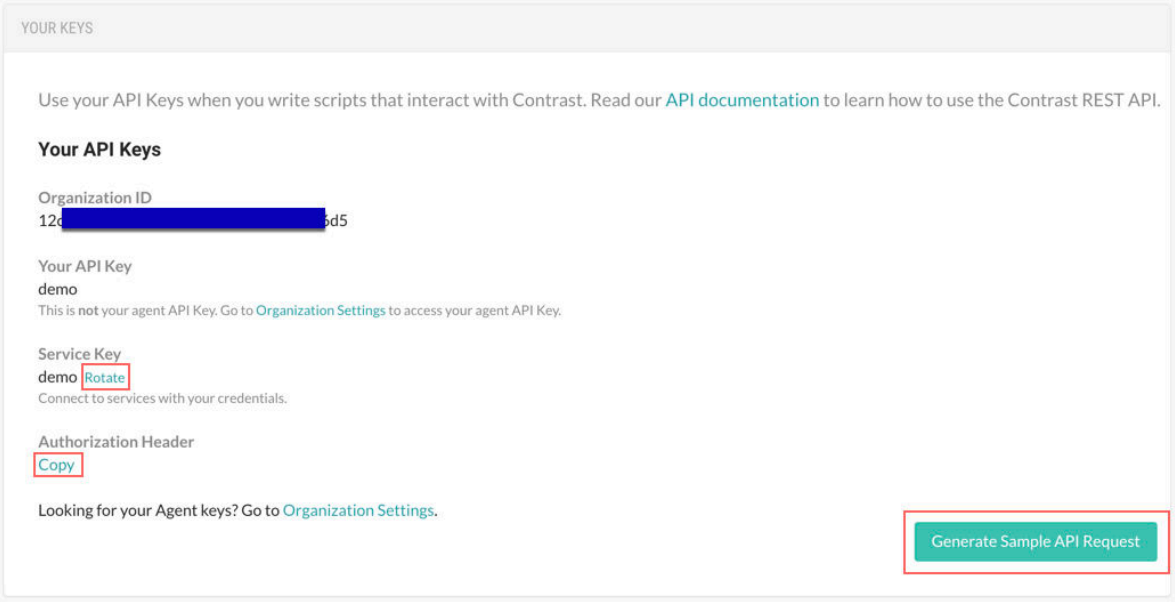
View your API keys

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:** Identities you as a valid 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".
   - **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 437) 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 743)
- Organization roles (page 744)

Applications

After an application has been instrumented (page 30), you can view your applications (page 438) in Contrast and explore these features:

- Route coverage (page 448)
- Session metadata (page 445)
- Flow maps (page 453)
- Application scoring (page 747)

NOTE
An Organization or Application Administrator will need to license your application (page 633) for these features to work.

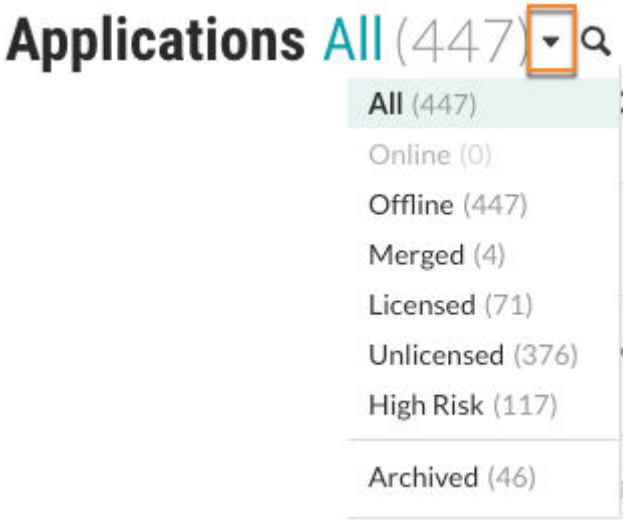
View applications

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.

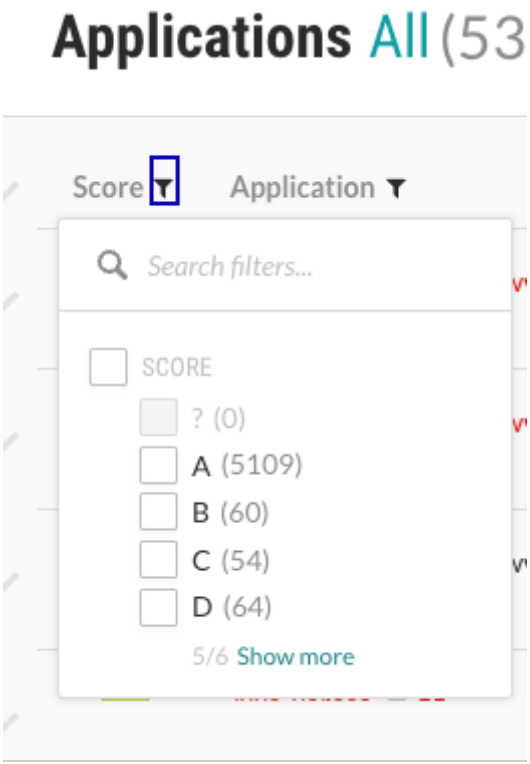
   Alternatively, to search for specific application by name, select the **magnifying glass** icon ( ) to search for them.



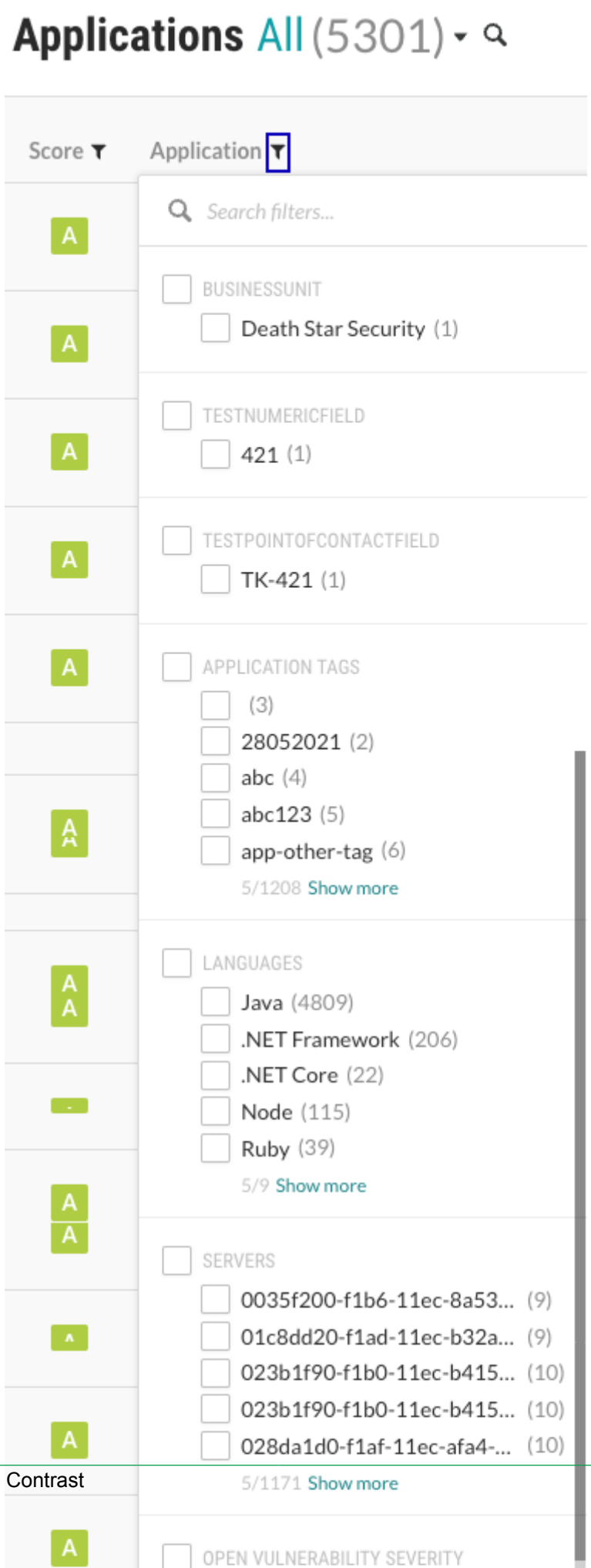
   The filters include:
   - **All:** Applications that you added to Contrast, excluding archived applications.
   - **Online:** Only applications whose agents have contacted Contrast within the last 5 minutes. Excludes archived applications.

- **Offline:** Only applications whose agents have had no contact with Contrast for more than 5 minutes.
  Excludes archived applications.
- **Merged:** Only applications that are 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.



5. To filter the list of applications, select the **filter** icon ( ) next to the Application column header. The filters include:
   - **Application metadata (if configured):** Custom 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.


Applications All (5301)
Score
Application
Search filters...
BUSINESSUNIT
Death Star Security (1)
TESTNUMERICFIELD
421 (1)
TESTPOINTOFCONTACTFIELD
TK-421 (1)
APPLICATION TAGS
(3)
28052021 (2)
abc (4)
abc123 (5)
app-other-tag (6)
5/1208 Show more
LANGUAGES
Java (4809)
.NET Framework (206)
.NET Core (22)
Node (115)
Ruby (39)
5/9 Show more
SERVERS
0035f200-f1b6-11ec-8a53... (9)
01c8dd20-f1ad-11ec-b32a... (9)
023b1f90-f1b0-11ec-b415... (10)
023b1f90-f1b0-11ec-b415... (10)
028da1d0-f1af-11ec-afa4-... (10)
5/1171 Show more
OPEN VULNERABILITY SEVERITY

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

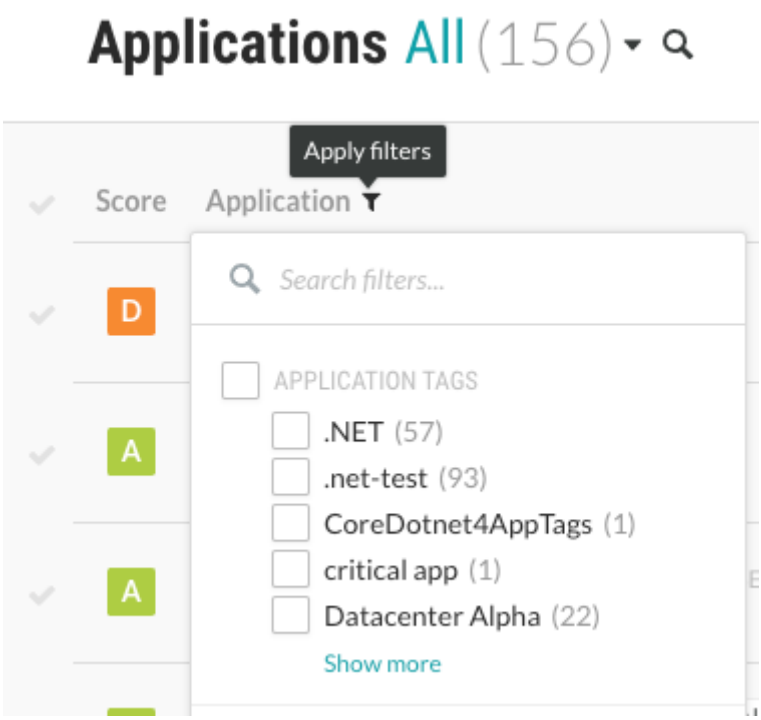
Use application tags to better organize applications and improve search functionality.

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.



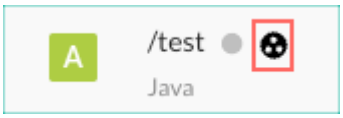
Merge and unmerge applications

Merging two or more applications creates a single application called a primary application and is a common operation for Organization Administrators responsible for bringing applications online.

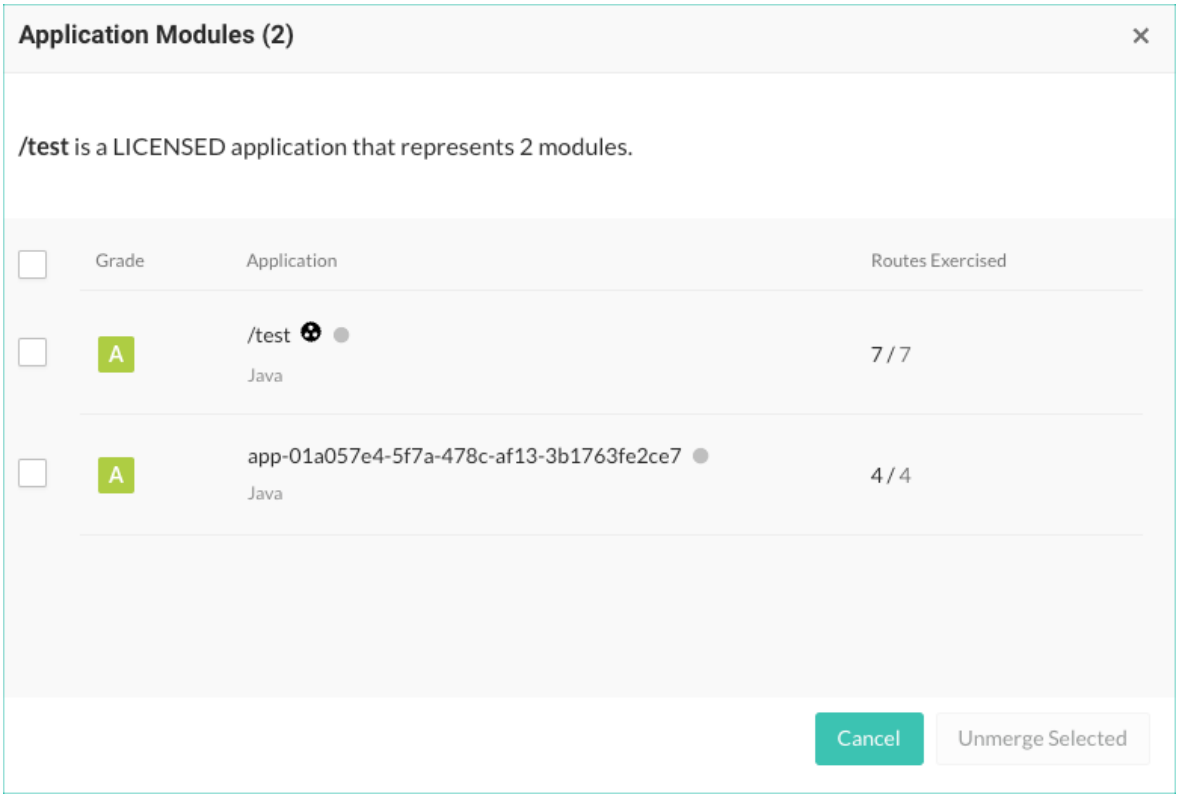
The main purpose of merging is to present a single application view for the purposes of scoring, discovered vulnerabilities, and remediation. Applications can be made up of module, which may show up individually in the application list. Merging also allows you to logically organize all of an application's modules into one entity in Contrast.

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 .



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

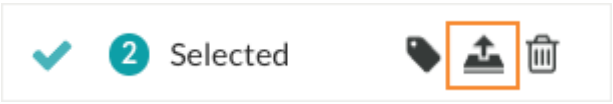
- Archiving an application doesn't free up a license in Contrast. To return a license back to the pool of available licenses is to archive and fully delete (page 444) the application.

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 444) it to make sure that all associated vulnerabilities, URLs and components are cleared properly.

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.

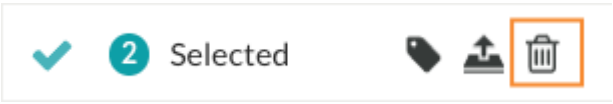
Licenses applied to applications permanently count towards the number of maximum allowable applications. Deleting a licensed application has no effect on the number of licenses you are allowed to apply to applications

Before you begin

- An Organization Administrator role is required.
- **Optional**: You can reset the application (page 444) 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 447) to your agent configuration file, the agent reports this information along with the rest of your standard vulnerability data to Contrast.

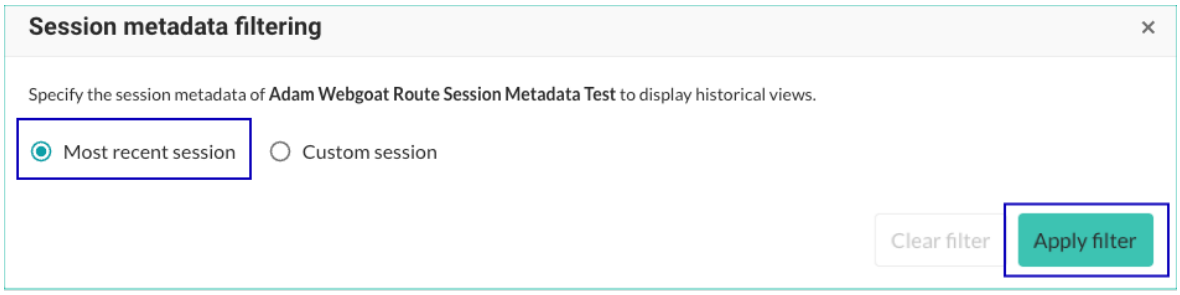
NOTE
This feature is currently not available to on-premises customers.

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.

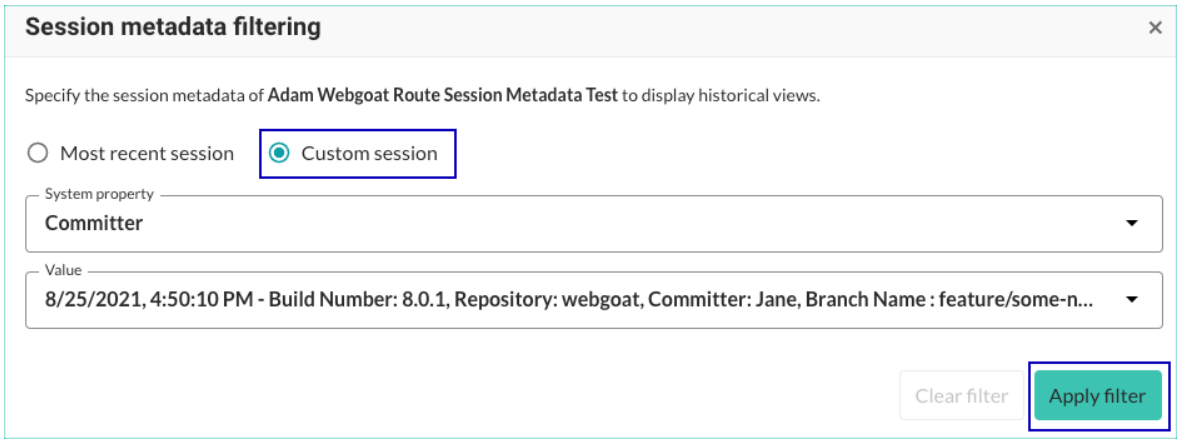
If you don't select a specific session metadata filter, the Session column in the Vulnerability list displays up to 10 of the values specified in the agent configuration file. This limit ensures that the Contrast web interface can display vulnerability and sink group data correctly.

Steps

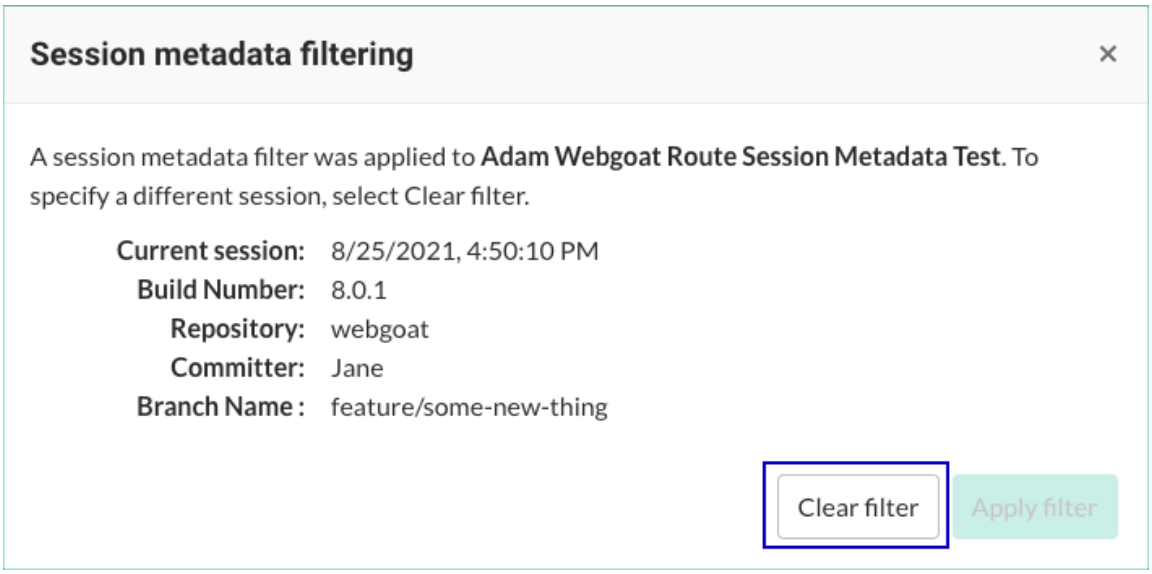
1. Select **Applications** in the header.
2. Select an application in the list.
3. Select either the **Vulnerabilities** tab or the **Route coverage** tab.
4. Select the Session metadata icon ( ) at the top of the list.
   The session metadata filters that you apply affect the Vulnerabilities and Route coverage lists.
5. Select the Session metadata icon ( ) at the top of the list.
   In the Vulnerabilities list, the Session column and the View by filter are hidden and the details of the most recent session display above the session metadata icon.



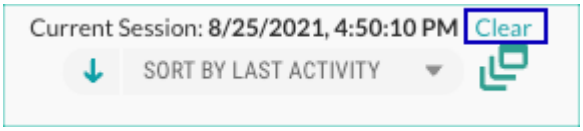
6. In the Vulnerabilities list, the Session column and the View by filter are hidden and the details of the most recent session display above the session metadata icon.
   a. In **System property,** select one of the displayed properties.
   b. In **Value**, start typing to find values for the selected system property.



7. To clear a session metadata filter, use one of these methods:
   - Select the Session metadata icon ( ). and select **Clear filter**.



- Select **Clear** next to the displayed metadata above the Session metadata icon ( ).



In the Vulnerabilities list, the Session column displays after you clear the session metadata filter.

Configure session metadata

To send session metadata for your application to Contrast, you must add the configuration settings 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	Value
Commit Hash	`commitHash`
Committer	`committer`
Branch Name	`branchName`
Git Tag	`gitTag`
Repository	`repository`
Test Run	`testRun`
Version	`version`
Build Number	`buildNumber`

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" />
```

- **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=$GIT_URL,buildN
umber=$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)`)

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

Contrast supports route discovery for these frameworks:

- **Java:** Jersey 2, Spring MVC 4-5, Struts 1, and Struts 2.
  Additionally, servlet route discovery is supported for: GlassFish 4.1, JBoss AS 4.2-7.1, Jetty 7.X-9.X, Tomcat 5.X-9.X, WebLogic 11g and 12c, WebSphere 8.5 and 9.0 and WildFly 8.1-18.0.
- **Scala:** Play 2.6, 2.7, 2.8
- **.NET Framework:** all currently supported frameworks (page 126)
- **.NET Core:** all currently supported frameworks (page 181)
- **Node.js:** all currently supported frameworks (page 230)
- **Python:** all currently supported frameworks (page 295)
- **Ruby:** all currently supported frameworks (page 355)
- **Go:** all currently supported frameworks (page 410)

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.

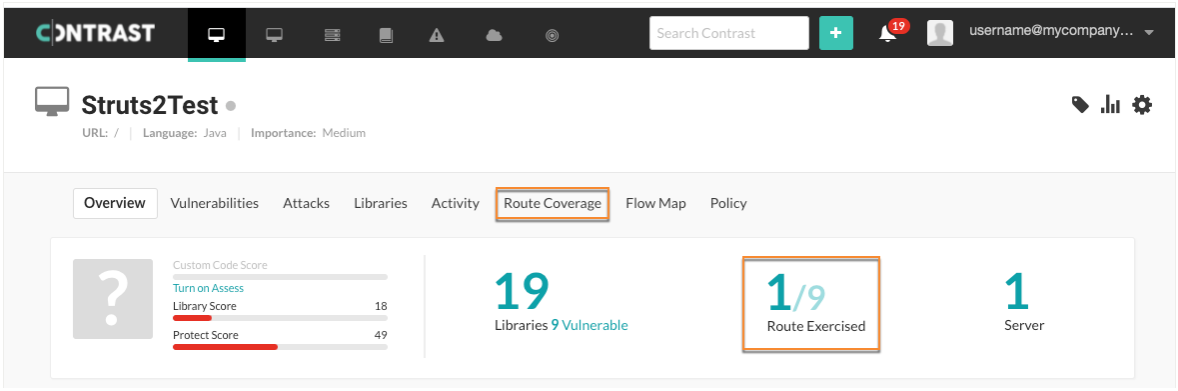
View route details

Route coverage (page 448) helps you understand how vulnerabilities map to your application's attack surface.

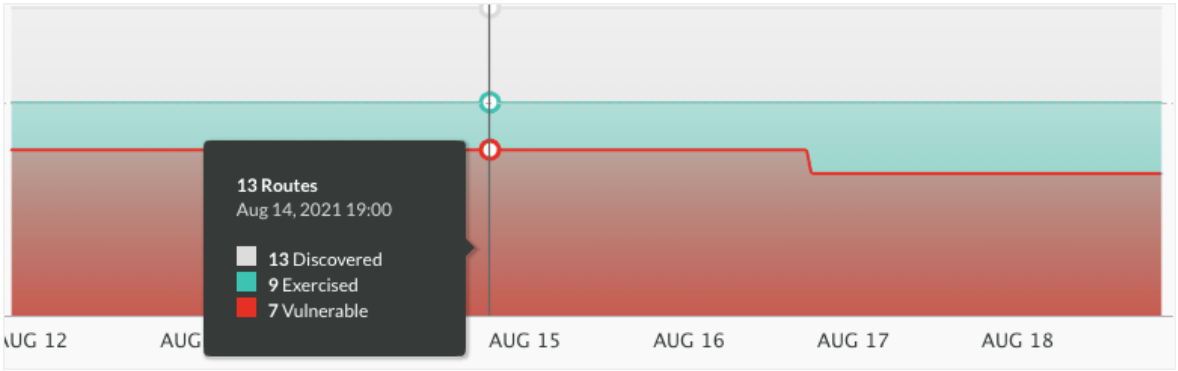
If you remove routes from the Route coverage list (steps 7 and 8) and they still exist when the server restarts or you exercise the application, Contrast includes them in the list again. To permanently exclude routes, select the **Exclude** icon ( ) at the end of the route's row.

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, hover over the chart to see details about the route during a specified time span.



The chart displays details about routes based on their status:

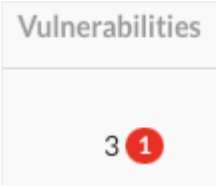
- Discovered by Contrast (but never exercised with the agent)
- Exercised with the Contrast agent
- Exercised and found to contain vulnerabilities

5. In the list, view additional details about each route.
   - **Route:** A route that Contrast identified or is tracking.
   - **Environment:** The environment for the server hosting the application: Development, QA, or Production.
   - **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.**

> **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.
   - **Application:** The application associated with the route.
   - **Last activity:** The activity time span for the route.
   - **Status**: The route status.
6. 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 the number in the Vulnerability column. A red warning mark indicates a critical vulnerability.



c. To view routes with a specific status only, select the triangle ( ) at the top of the page and select a filter:
   - **All:** Shows all routes that are not excluded.
   - **Exercised:** Shows only routes that are exercised.
   - **Not Exercised:** Shows routes that Contrast discovered but are not exercised.
   - **Vulnerable:** Shows only routes that have a vulnerability associated with them.
   - **Excluded:** Shows only routes that you excluded from the application scoring calculations.

d. To view routes in specific environments, select the Filter icon ( ) next to the Environment column.

e. To view routes for specific modules in a merged application, select the Filter icon ( ) next to the Application column.

f. 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.

7. 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, click **Delete**.
8. 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, click **Delete**.
9. 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 452)
- Include routes (page 452)

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 451) 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

- An Admin role is required.
- Identify the routes that you want to exclude.

Steps

1. Select **Applications** from the header.
2. Select an application name.
3. Select the **Route Coverage** tab.
4. Exclude one or more routes:
   a. To exclude a single route, hover over the end of the row and select the **Exclude** icon ( )
   b. 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.

   c. Confirm the exclusion in the Exclude route window.

   The status for the selected routes changes to **Excluded**.

Include routes

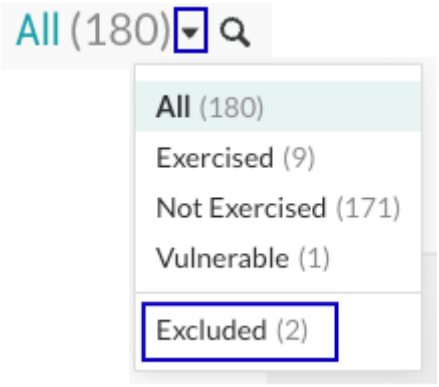
You can include routes (page 451) that you previously excluded. On the Route coverage page, excluded routes have a status of **Excluded**.

Before you begin

- An Admin role is required.
- Identify the excluded routes that you want to include.

Steps

1. Select **Applications** in the header.
2. Select an application name.
3. Select the **Route coverage** tab.
4. 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.

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 454), 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 454).

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 636) 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 454).

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 454) 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

In Contrast Scan, you can:

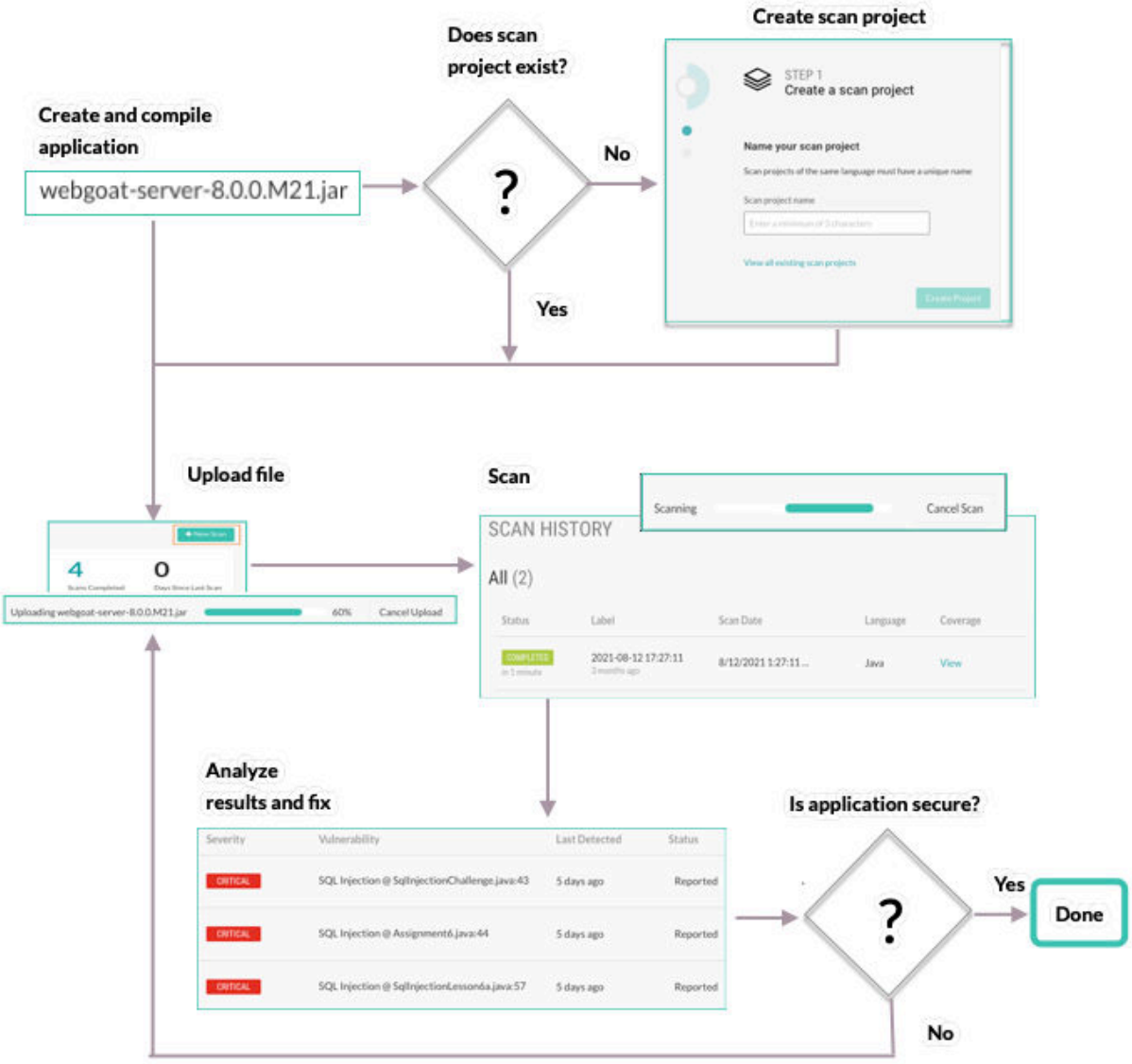
- Create a scan project (page 459)
- Monitor scans (page 461)
- Analyze scan results (page 468)
- Start a new scan (page 460)
- Cancel a scan (page 461)
- Change scan settings (page 479)

Scan process

This section provides details about the workflow for using Contrast Scan as well as the process Scan uses when analyzing your code.

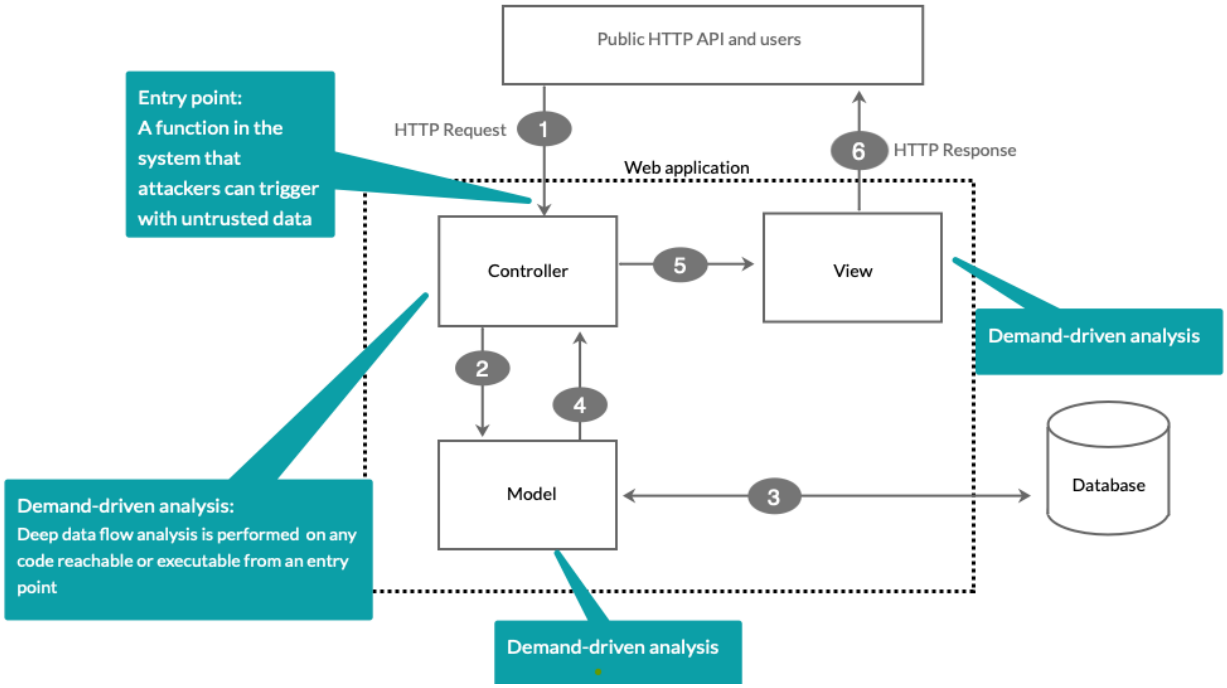
Scan workflow

This diagram illustrates the Scan workflow that you follow.

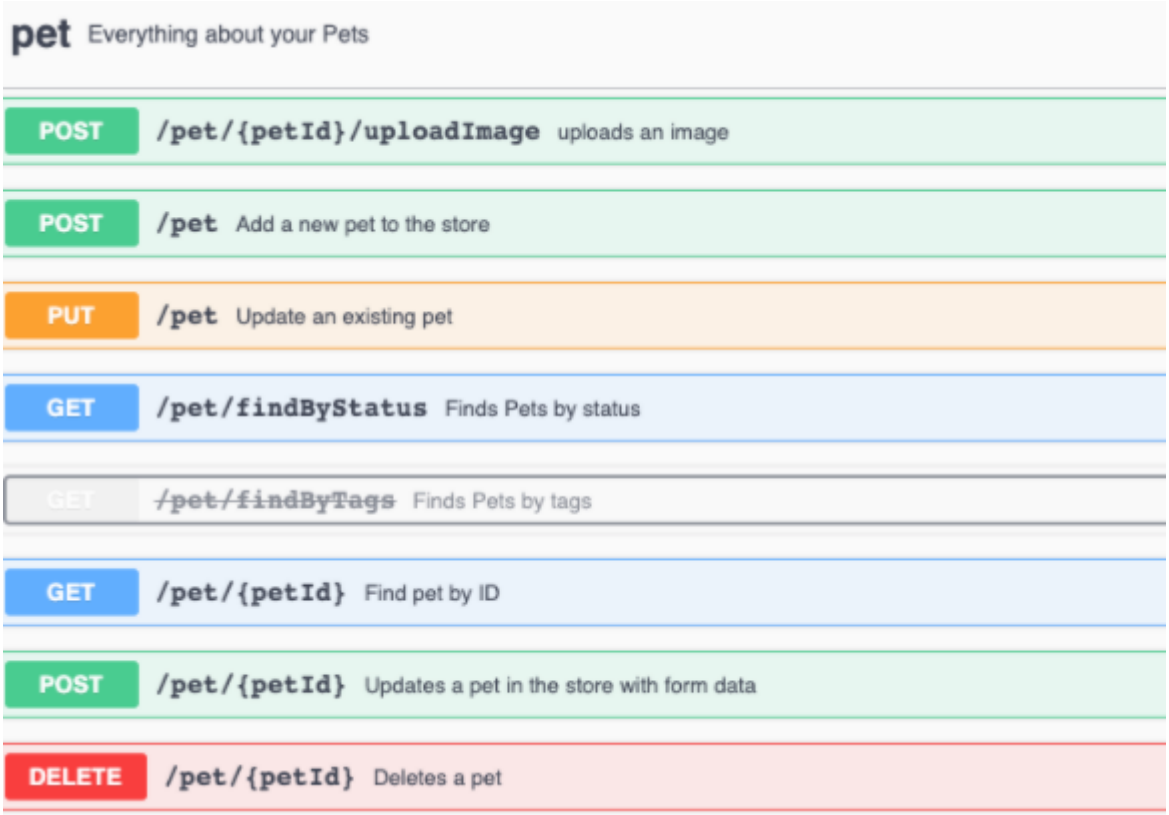


Application view

This diagram shows how Scan views a typical web application for the purpose of data flow analysis.



Scan looks at data entry points to find code to scan. Here are some examples of typical data entry points that Scan examines:



Contrast Scan supported languages

Contrast Scan supports these languages:

Language	File types
Java (for example: J2EE, JSP, and Spring MVC)	JAR and WAR files
JavaScript • Angular version 8 or later • JQuery (all versions) • React version 16 or later • Vue.JS 2 or later • Typescript 4.2 or later	JS and ZIP files
Python 3.8 or later The only supported framework is Django.	ZIP files

Scan package preparation

To get the best results from a scan, consider these best practices before you upload packages.

Artifact types

- For Java, upload either a WAR or JAR package.
- For JavaScript, upload either a JS or ZIP package.
- For Python, upload a ZIP package.
  Python is supported with the Contrast local scan engine (page 462) only.
  The only supported framework is Django.

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.

Scan 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.

Create a scan project

Scan projects are containers for artifacts that you want to scan.

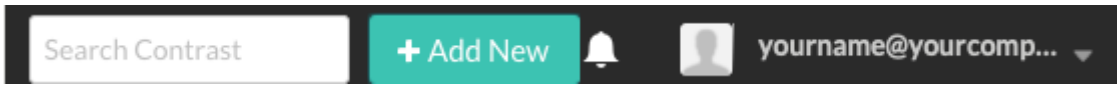
Creating a scan project starts the first scan for an uploaded file.

Before you begin

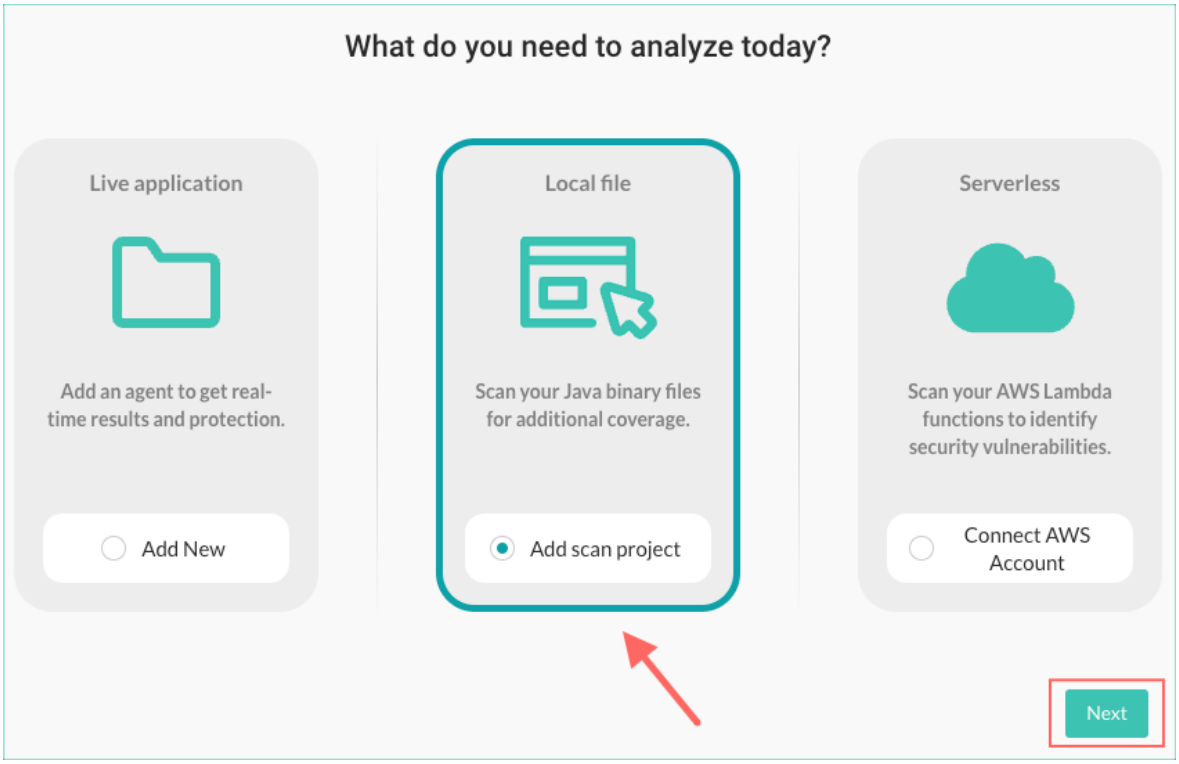
- Identify the artifact that you want to upload for scanning.
  Scan supports different files types for each programming language. For example, for Java,upload a JAR or WAR file.

Steps

1. In the Contrast web interface, select **Add New** in the top right corner.



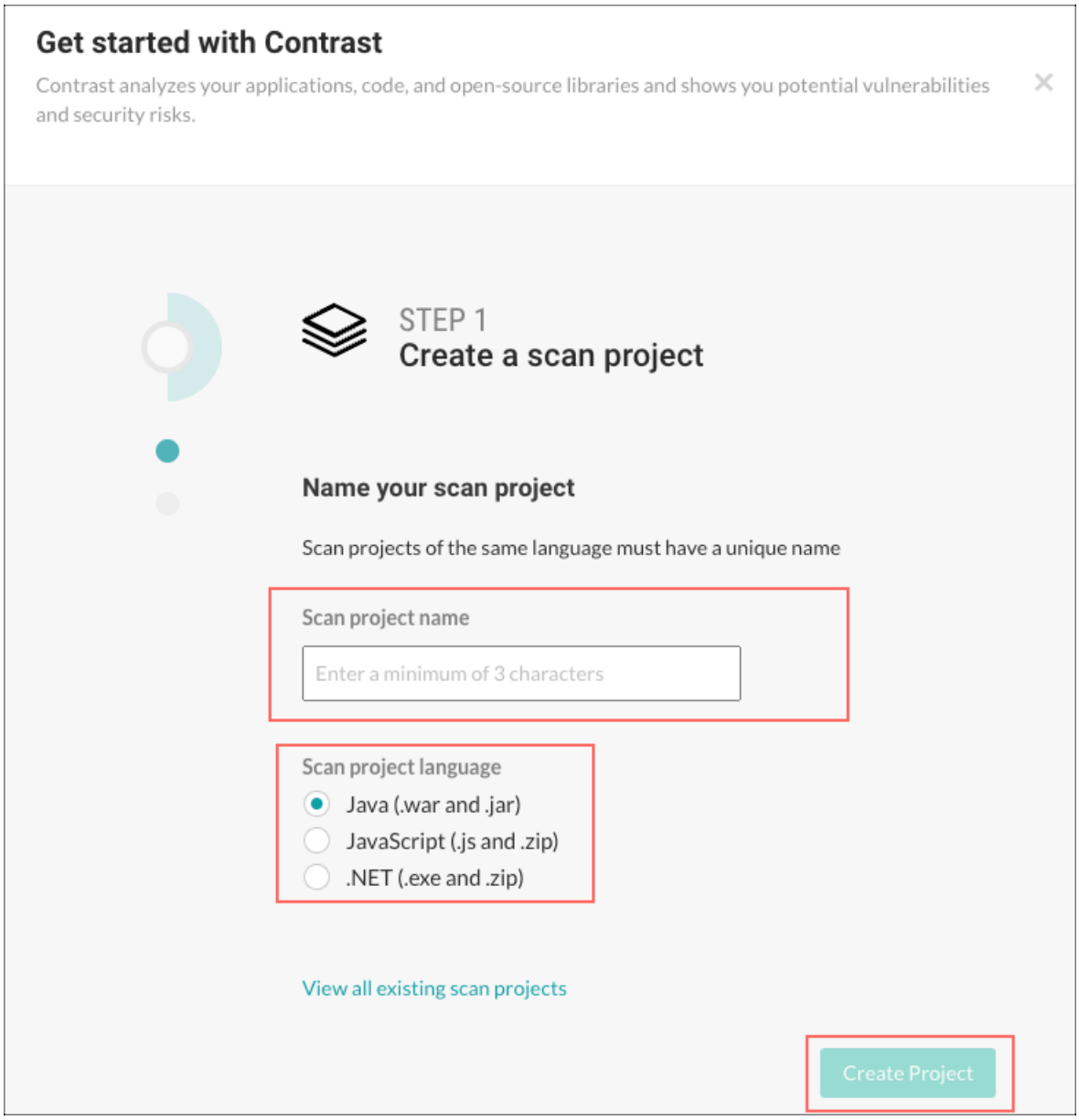
   This option is available when Contrast Scan is enabled. I this opiton is not enabled, contact Support.
2. In Local File, select **Add scan project**. Then, select **Next**.



3. 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 artifact. For example, if your artifact is `webgoat.jar`, name your project `webgoat` or `webgoat.jar`.

4. Select the language for the files that the project will contain.



5. Select **Create project**.

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 459).

Steps

1. Select **Scans** in the header.

2. Select a scan project.
3. Select **New Scan**.

4. From the displayed window, select a file 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.
5. Monitor (page 461) 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.

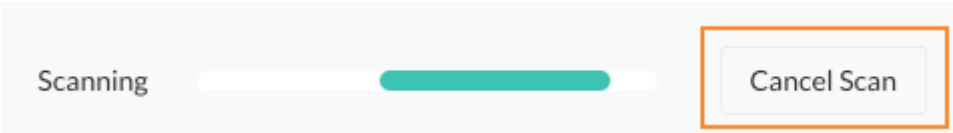
Before you begin

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.

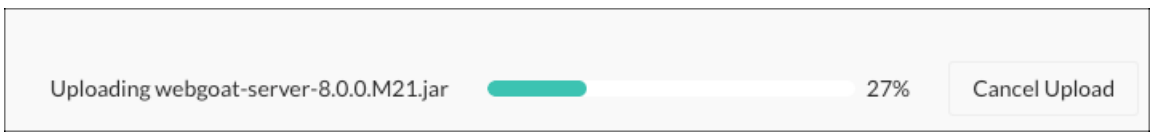
Monitor scans

After you start a scan, you can monitor the progress of the file upload and the scan from any tab under Scans.

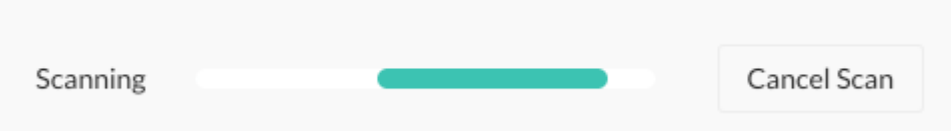
You can view a history of the scans in a selected project under Scan history in the Overview tab.

Steps

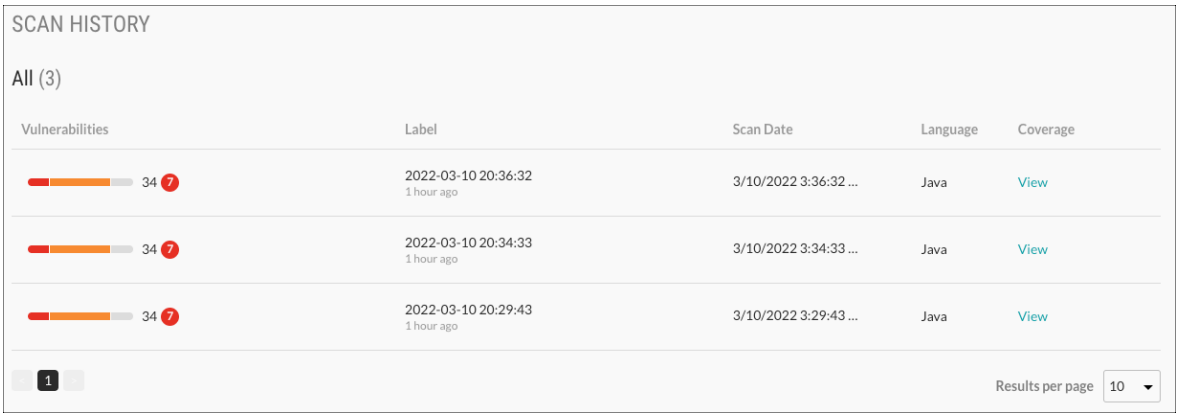
1. Select **Scans** in the header.
   The Scans page shows these scan details:
   - The grade score for each scan project.
   - The names of the scan projects.
   - The number and status of open vulnerabilities.
   - The time the last scan completed.
2. Select a scan project.
3. Start a scan (page 460).
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.



6. To see additional details about a scan, select a scan label or, under the Coverage. column, select **View**.

Contrast Scan local engine

The Contrast Scan local engine lets you scan your application using Docker CLI commands or 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 and in a Docker container.

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 464):

Variable	Description
CONTRAST__API__PROXY__ENABLE	Enables proxy settings.
CONTRAST__API__PROXY__URL	**Required**. The URL for the proxy server (for example, `http://host:port`)
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

Scan process

To use the Scan local engine:

1. Decide how you want to run the local scan:
   - Download the Docker container (page 463) that contains the local scanner application from Contrast Security.
   - Download the Scan local engine JAR file (page 463).
2. Decide if you want to use a proxy server for uploading results.
3. Run the scan on a local system (page 464).
4. View results (page 468) in the Contrast web interface.

Download the Docker container

To get started with using the Contrast local scanner, download the Docker container from the GitHub repository for Contrast.

If you prefer to use a Java JAR file instead of the Docker container, download the JAR fie. (page 463)

Before you begin

- Install the Docker CLI.
- Configure Docker to use 12 GB of memory.
- Get a Personal Access Token (PAT) from Contrast Support.

Steps

1. Configure this variable on your local system:

   ```
   export CONTRAST_PAT=<ContrastProvidedPAT>
   ```

   Replace `<ContrastProvidedPAT>` with the PAT that Contrast Support gives you
2. Log in and download the Docker container with these commands:

   ```
   docker login ghcr.io/contrast-security-inc -u local-scanner -
   p $CONTRAST_PAT
   docker pull ghcr.io/contrast-security-inc/contrast-sast-scanner-
   java:latest
   ```

Download the Java JAR file

To download the Java JAR file for the Contrast local scan engine, use one of these options:

- Download from Maven.
- Download with a curl command.

Before you begin

- Get a Personal Access Token (PAT) from Contrast Support.

Download from Maven

1. Create a Project Object Model (POM) file with this information:

   ```
   <server>
     <username>local-scanner</username>
     <password>ContrastProvidedPAT</password>
     <id>github</id>
   ```

```
</server>

<repository>
  <id>github</id>
  <name>github</name>
  <url>https://maven.pkg.github.com/Contrast-Security-Inc/sast-local-scan-
runner</url>
</repository>

<dependency>
    <groupId>com.contrastsecurity</groupId>
    <artifactId>sast-local-scan-runner</artifactId>
    <version>latest</version>
</dependency>
```

Replace `ContastProvidedPAT` with the PAT that Contrast Support gives you.

2. Right-click the POM file and select **Run As > Maven Install**. or use the `mvn install` command.

Download with curl commands

- Use this curl command to download the Java JAR file:

```
curl -L -H 'Accept: application/vnd.github.v3.raw' -
s https://<ContrastProvidedPAT>@maven.pkg.github.com/Contrast-
Security-Inc/sast-local-scan-runner/com.contrastsecurity.sast-local-scan-
runner/<VersionNumber>/sast-local-scan-runner-<version number>.jar -
o scanner.jar
```

  - Replace <`ContrastProvidedPAT`> with the PAT that Contrast Support gives you.
  - Replace <`VersionNumber`> with the latest version (do not use `latest` in the curl command). Ask your Contrast representative for the latest version number.

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 `/api/sast` 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, `http://host:port`)
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
UMBRELLA_LAUNCHER_JAVA_PATH	No	Path for the umbrella Java scanner jar (for example, `/app/contrast-scan-java-cli.jar`)

Run local scan

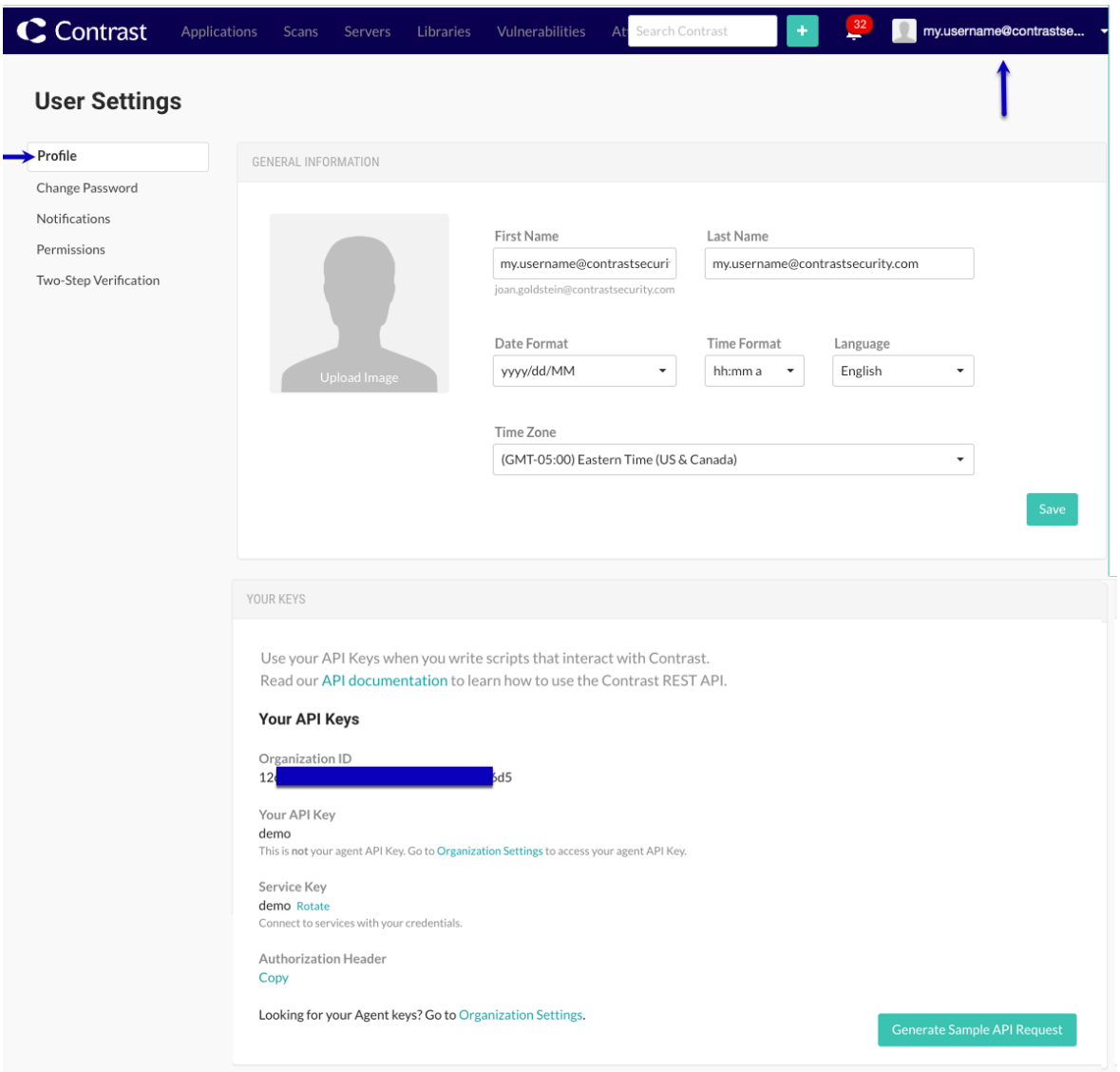
The Contrast Scan local engine is available as a Docker container or as a Java JAR file. It requires the location of a build artifact so it can process the scan.

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`.
- If you plan to use the Java JAR version of the Scan local engine, ensure that this software is installed on your system:
  - Java 11
  - If you are scanning JavaScript project files, Semgrep App version 0.114.0
- Internet access to let the local scanner upload scan results and scanner output to Contrast.
- Verify that you have 1 CPU available (the Contrast Scan local engine is single threaded) and 12 GB of RAM.

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 462)

```
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 `/api/sast` at the end of the URL.
- 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 to the folder where the file you want to scan is located.
  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:
   - If you are using the **Docker version** of the Scan local engine, start the scan with a command similar to this one:

```
docker run -v $LOCAL_ARTIFACT_LOCATION:/app/artifacts \
    -v $LOCAL_OUTPUT_LOCATION:/app/results \
    -e CONTRAST__API__URL=${CONTRAST__API__URL} \
    -e CONTRAST__API__USER_NAME=${CONTRAST__API__USER_NAME} \
    -e CONTRAST__API__API_KEY=${CONTRAST__API__API_KEY}  \
    -e CONTRAST__API__SERVICE_KEY=${CONTRAST__API__SERVICE_KEY}  \
    -e CONTRAST__API__ORGANIZATION=${CONTRAST__API__ORGANIZATION} \
    ghcr.io/contrast-security-inc/contrast-sast-scanner-java:latest \
    --project-name "<ProjectName>-${CONTRAST__API__ORGANIZATION}" \
    --label "<LabelName>" \
    /app/artifacts/<FileName> \
    -o $LOCAL_TARGET_OUTPUT_LOCATION/results.sarif
```

     - Replace <ProjectName> with the name of a Scan project.
     - Replace <LabelName> with a label for the scan. For example, you could specify a build number as: "build:1.0.1".
     - Replace <FileName> with the name of the file that you want to scan. This file must reside in the directory that you specified for the $LOCAL_ARTIFACT_LOCATION.
   - If you are using the **Java JAR file** to run the Scan local engine, 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 that you want to scan. You can also specify folders.
     - 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.

Command options

Use any of these command options with the Docker container or the Java JAR file:

Option	Description
`-o, --output-results`	Specifies the location for output results. If you don't specify a location, the local scanner writes the results to the current working directory.
`-V, --version`	Prints version information

Option	Description
`--project-name <ProjectName>`	The name of a scan project. 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. If the project name includes spaces, enclose the name in double quotes ("). For example: "My Scan Project". This option is required if you don't use a project ID.
`--project-ID`	The ID for an existing project. This option is required if you don't use a project name.
`--label <label>`	A label for the current scan.

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
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

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

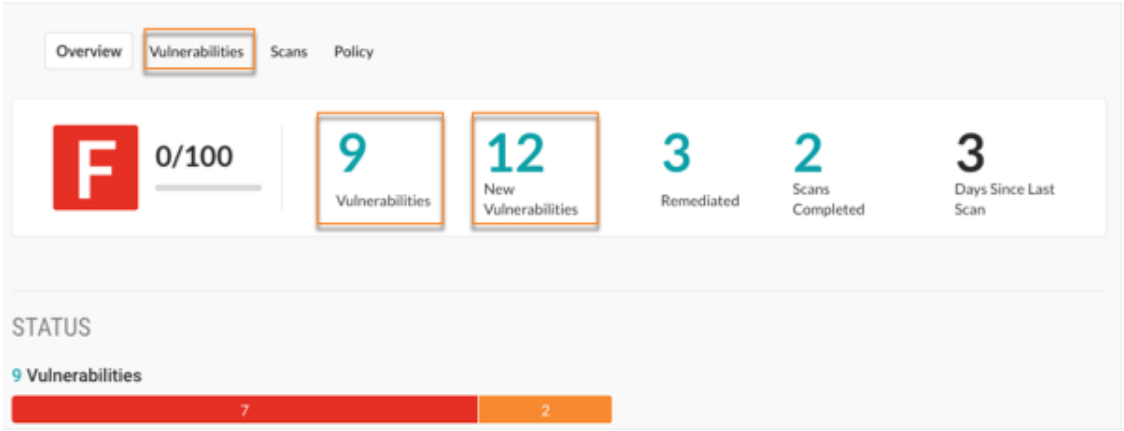
A scan observes the data flow in an application and reports vulnerabilities that it discovers.

After you analyze the results, update your code and run the scan again to verify the vulnerability is fixed.

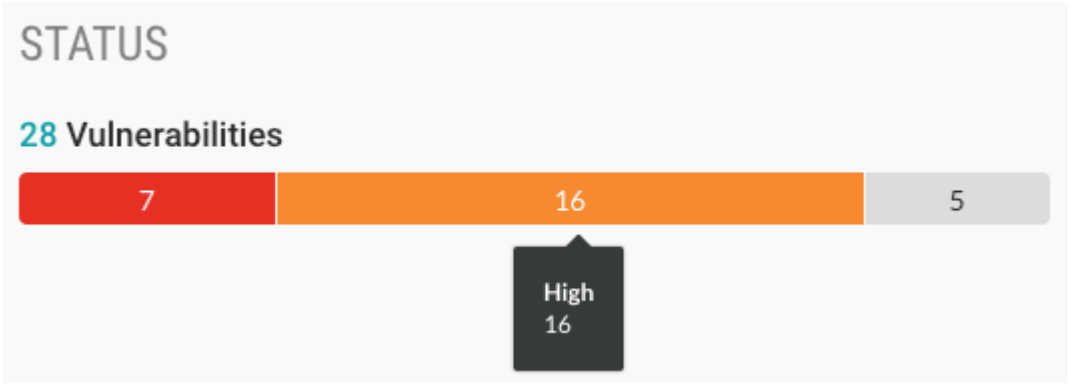
Steps

To find information on vulnerabilities after a scan completes:

1. Select **Scans** in the header.
   The Scans page shows a list of scan projects.
2. To view a list of discovered vulnerabilities and their severity:
   a. Select a scan project.
   b. In the Overview tab, click a vulnerability number or select the **Vulnerabilities** tab.



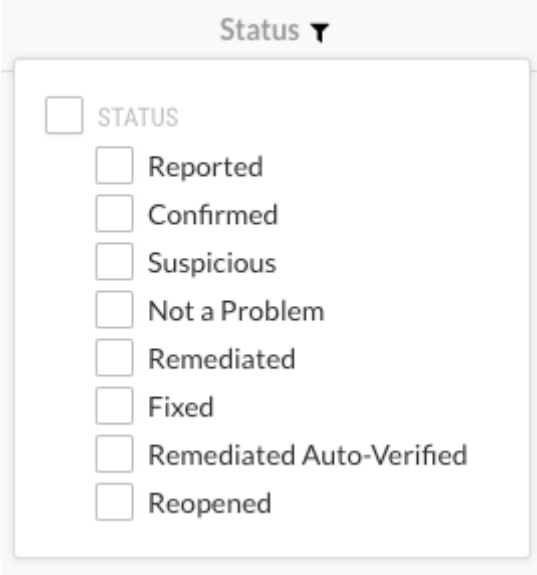
Alternatively, to see vulnerabilities with a specific status, under Status, select a section in the Vulnerability bar.



3. On the Vulnerabilities tab, to sort the vulnerabilities by status or severity, select the **Filter** icon ( ) next to the Severity or Status columns and select one or more statuses.

Severity filters:

Status filters:



To clear a filter, select **Clear** next to the Severity or Status column.

4. To view more information about a specific vulnerability, in the Vulnerabilities tab, select the vulnerability.
   - 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.
5. 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.
6. To view suggestions for fixing the code, select the **How to fix** tab.
7. To view additional details about the vulnerability, select the **Notes** tab:
   - 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
8. To change the status of a vulnerability, in the Status column, select the current status for a vulnerability and select a vulnerability status (page 535).

View scan details

Scan details include:

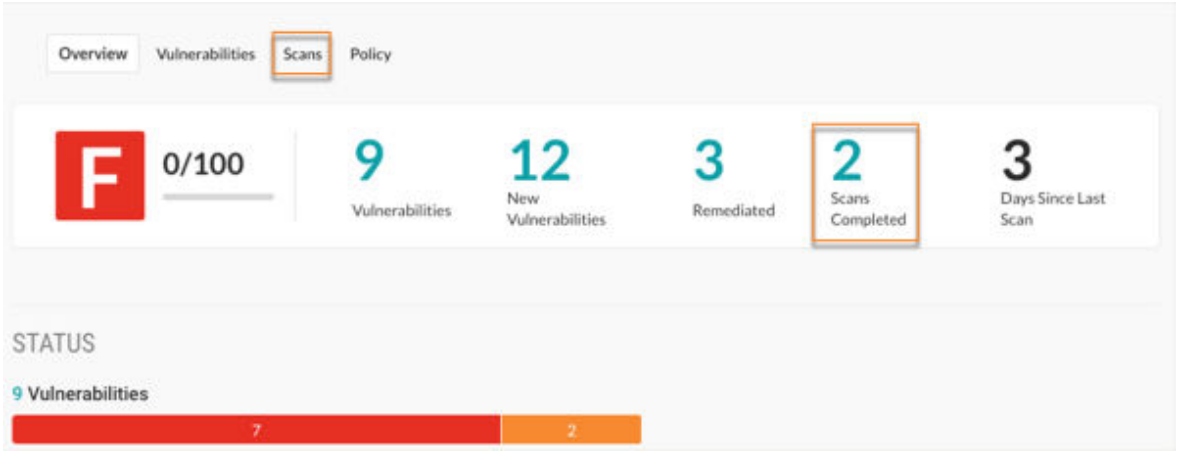
- A summary of the scan results
- Scan coverage details

Before you begin

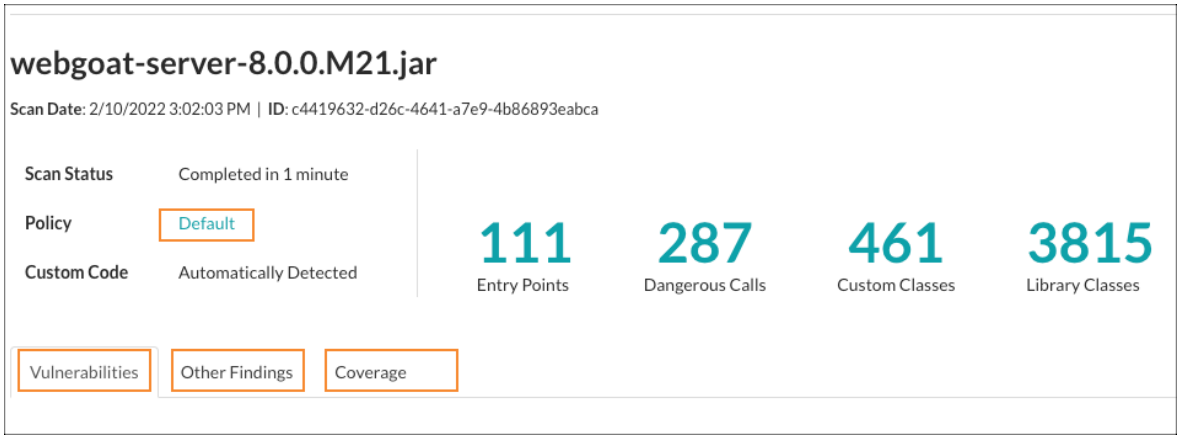
- Identify the scan project that contains the scan you want to view.

Steps

1. In the header, select **Scans** .
2. Select a scan project.
3. Select the number for **Scans completed** or select the **Scans** tab.



4. Under Scan history, select a scan or select **View** in the Coverage column.
5. View the details:



- At the top of the list, view a summary of scan details. To view the rules that the scan used, next to Policy, select **Default**.
- In the Vulnerabilities tab, view the vulnerabilities that the scan found.
  Contrast has high confidence that the vulnerabilities in this list require remediation.
- In the Other Findings tab, 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.
- In the Coverage tab, view the classes that the scan included and excluded.

Download scan results

After a scan completes, you can download the results to a Static Analysis Results Interchange Format (SARIF) file. This type of file is is a standard, JSON-based format for the output of static analysis data.

To optimize storage usage, downloads are available for up to five days after the scan completes. No download option is available for older scans.

Before you begin

- Identify the scan whose results you want to download.

Steps

1. Select scan in the header.
2. Select a scan project.
3. To download scan results from Scan history, hover over the end of a row for a scan and select the download icon ( ).

4. To download scan results from the scan details page:
   a. Under Scan history, select a scan or select **View** in the Coverage column for the scan.
   b. In either the Overview or Vulnerabilties tab, select the download icon ( ).

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:

```
"tool" : {
  "driver" : {
    "name" : "Contrast Scan",
    "organization" : "Contrast Security, Inc.",
    "version" : "pkg: 2.0.0-SNAPSHOT, engine: 2.0.0-SNAPSHOT, policy: 2.0.0-SNAPSHOT",
    "informationUri" : "https://www.contrastsecurity.com"
  }
```

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.

```
  "artifacts" : [ ],
  "results" : [ {
    "ruleId" : "sql-injection",
    "level" : "error",
    "message" : {
      "text" : "sql-injection in User.fetch() reachable from LoginController.login()"
    },
    "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" : { }
        }
      }
    } ],
    "partialFingerprints" : {
      "GITHUB_SOURCECODE_LSH/v1" :
      "e86a7a0c38715f4a:1-158823731d54d12e:1-6b692237789f9a2a:1-64ed8e6526b79cf4:1-13db8c734146ff90:1"
    },
    "codeFlows" : [ {
      "message" : {
        "text" : "Untrusted data flow from LoginController.java:19 to User.java:49 via variable `query`"
      },
      "threadFlows" : [ {
        "locations" : [ {
          "location" : {
            "physicalLocation" : {
              "artifactLocation" : {
                "uri" : "com/scalesec/vulnado/LoginController.java"
              },
              "region" : {
                "startLine" : 19,
                "snippet" : {
                  "rendered" : {
                    "text" : "@CrossOrigin(origins={\"*\"})\n@RequestMapping(value={\"/login\"}, method={\"POST\"},
                    produces={\"application/json\"}, consumes={\"application/json\"})\nLoginResponse login(**@RequestBody
                    LoginRequest input**) {\n  ...\n}"
                  }
                },
                "properties" : {
                  "ir" : [ "@CrossOrigin(origins={\"*\"})\n@RequestMapping(value={\"/login\"}, method={\"POST\"},
                  produces={\"application/json\"}, consumes={\"application/json\"})\nLoginResponse login(**@RequestBody
                  LoginRequest input**) {", "  ...", "}" ]
                }
```


- **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:

```
"threadFlows" : [ {
  "locations" : [ {
    "location" : {
      "physicalLocation" : {
        "artifactLocation" : {
          "uri" : "com/scalesec/vulnado/LoginController.java"
        },
        "region" : {
          "startLine" : 19,
          "snippet" : {
            "rendered" : {
              "text" : "@CrossOrigin(origins={\"*\"})\n@RequestMapping(value={\"/login\"}, method={\"POST\"},
              produces={\"application/json\"}, consumes={\"application/json\"})\nLoginResponse login(**@RequestBody
              LoginRequest input**) {\n  ...\n}"
            }
          },
          "properties" : {
            "ir" : [ "@CrossOrigin(origins={\"*\"})\n@RequestMapping(value={\"/login\"}, method={\"POST\"},
            produces={\"application/json\"}, consumes={\"application/json\"})\nLoginResponse login(**@RequestBody
            LoginRequest input**) {", "  ...", "}" ]
          }
        }
      },
      "logicalLocations" : [ {
        "name" : "LoginController.login()",
        "fullyQualifiedName" : "com.scalesec.vulnado.LoginController.login(com.scalesec.vulnado.LoginRequest)"
      } ]
    }
  }, {
    "location" : {
      "physicalLocation" : {
        "artifactLocation" : {
          "uri" : "com/scalesec/vulnado/LoginController.java"
        },
        "region" : {
          "startLine" : 20,
          "snippet" : {
            "rendered" : {
              "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}"
            }
          },
          "properties" : {
            "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" ]
          }
        }
      },
      "logicalLocations" : [ {
        "name" : "LoginController.login()",
        "fullyQualifiedName" : "com.scalesec.vulnado.LoginController.login(com.scalesec.vulnado.LoginRequest)"
      } ],
      "message" : {
        "text" : "un"
      }
    },
    "state" : {
      "un" : {
        "text" : "tainted",
        "properties" : {
          "taintTags" : [ "untrusted", "cross-site" ]
        }
```


- **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              "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              }
```

- **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
          StringBuilder();    // Java line 47\n  $stack1 = \"select * from users where
          username = '\";    // Java line 47\n  $stack0 = $stack0.append($stack1);    //
          Java line 47\n  $stack0 = $stack0.append(un);    // Java line 47\n  $stack1 =
          \"' limit 1\";    // Java line 47\n  $stack0 = $stack0.append($stack1);    //
          Java line 47\n  query = $stack0.toString();    // Java line 47\n  ...\n}"
        }
      },
      "message" : {
        "text" : "Propagation event of untrusted data occurred at 'User.java:47' in the
        method 'User.fetch()' to variable 'query'"
      },
      "properties" : {
        "ir" : [ "$stack0 = new java.lang.StringBuilder;    // Java line 47", "$stack1 =
        \"select * from users where username = '\";    // Java line 47", "$stack0 =
        virtualinvoke $stack0.<java.lang.StringBuilder: java.lang.StringBuilder append
        (java.lang.String)>($stack1);    // Java line 47", "$stack0 = virtualinvoke
        $stack0.<java.lang.StringBuilder: java.lang.StringBuilder append(java.lang.String)>
        (un);    // Java line 47", "$stack1 = \"' limit 1\";    // Java line 47", "$stack0
        = virtualinvoke $stack0.<java.lang.StringBuilder: java.lang.StringBuilder append
        (java.lang.String)>($stack1);    // Java line 47", "query = virtualinvoke $stack0.
        <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.

```
"scannedData" : {
  "scannedBodyClasses" : [ "com.scalesec.vulnado.ServerError",
    "com.scalesec.vulnado.LoginResponse",
    "org.springframework.lang.UsesSunMisc",
    "com.scalesec.vulnado.Postgres",
    "com.scalesec.vulnado.LinksController",
    "com.scalesec.vulnado.BadRequest",
    "com.scalesec.vulnado.Unauthorized",
    "com.scalesec.vulnado.Comment",
    "org.springframework.lang.NonNullFields",
    "org.springframework.lang.NonNull",
    "org.springframework.lang.UsesJava7",
    "org.springframework.lang.UsesJava8",
    "com.scalesec.vulnado.CowController",
    "com.scalesec.vulnado.LoginController",
    "org.springframework.lang.NonNullApi",
    "org.springframework.lang.Nullable",
    "org.springframework.lang.UsesSunHttpServer",
    "com.scalesec.vulnado.CommentsController",
    "com.scalesec.vulnado.Cowsay",
    "com.scalesec.vulnado.VulnadoApplication",
    "com.scalesec.vulnado.LoginRequest",
    "com.scalesec.vulnado.LinkLister",
    "com.scalesec.vulnado.User",
    "java.util.Optional",
    "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.

```
      "routesDiscovered" : [ {
        "routeSignature" : "com.scalesec.vulnado.LoginController.login(com.scalesec.vulnado.LoginRequest)"
      }, {
        "routeSignature" : "com.scalesec.vulnado.CowController.cowsay(java.lang.String)"
      }, {
        "routeSignature" : "com.scalesec.vulnado.CowController.cowsay2(java.lang.String)"
      }, {
        "routeSignature" : "com.scalesec.vulnado.LinksController.links(java.lang.String)"
      }, {
        "routeSignature" : "com.scalesec.vulnado.LinksController.linksV2(java.lang.String)"
      }, {
        "routeSignature" : "com.scalesec.vulnado.CommentsController.comments(java.lang.String)"
      }, {
        "routeSignature" : "com.scalesec.vulnado.CommentsController.createComment(java.lang.String,com.scalesec.vulnado.CommentRequest)"
      }, {
        "routeSignature" : "com.scalesec.vulnado.CommentsController.deleteComment(java.lang.String,java.lang.String)"
      } ]
    },
    "invocations" : [ {
      "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",
      "toolExecutionNotifications" : [ ],
      "executionSuccessful" : true
```


View scan policies

View policies to see which vulnerabilities Contrast looks for in your code.

At this time, editing or adding polices is not supported.

1. Select **Scans** in the header.
2. Select a scan project.
3. Select **Policy**.
   The policy list displays the policies used for scans.
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.

Before you begin

- An Admin role is required.

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. 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 518) 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 593) 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 i -g @contrast/contrast-cli@latest
```

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-cli  --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:

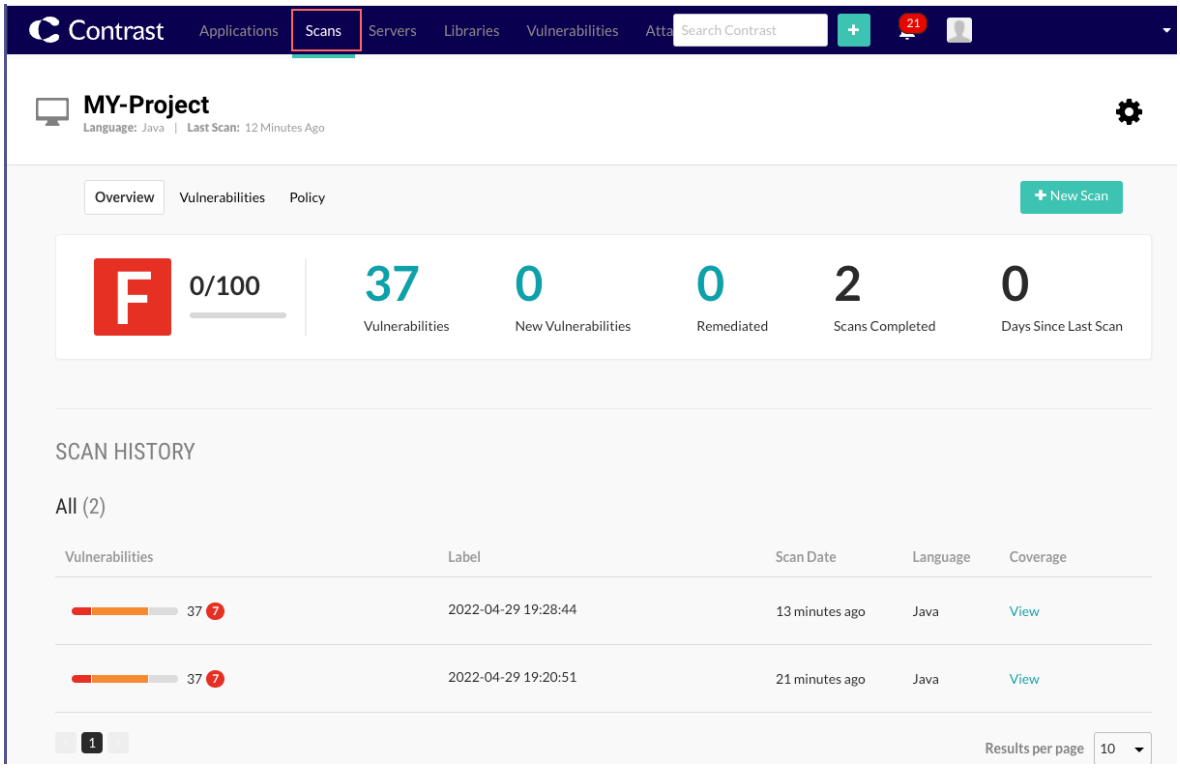
```
project created ID is 788f9734-b933-4f05-b391-c130931baf88
 Uploaded file successfully.
 Response:  {
  id: '5091d134-93ea-4873-8110-8cf99d14606e',
  organizationId: '74f4cd04-6ca9-4eb7-a7a7-78909c2101cc',
  projectId: '788f9734-b933-4f05-b391-c130931baf88',
  filename: 'param.war',
  createdTime: '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.



Examples

- Scan integration with GitHub (page 482)
- Scan integration with Jenkins (page 483)

Example: Scan integration with GitHub

Review the Scan integration steps (page 480) before you integrate Contrast Scan with GitHub.

This example shows how to set up a GitHub workflow.

```
- name: Set up contrast-cli
2 run: |
3 npm i -g @contrast/contrast-cli@0.0.29
4- name: Scan file
5 env:
6 CT_API_KEY: ${{ secrets.CONTRAST__API__API_KEY }}
7 CT_AUTH_TOKEN: ${{ secrets.CONTRAST__API__AUTH_TOKEN }}
8 ORG_ID: ${{ secrets.CONTRAST__API__ORGANIZATION_ID }}
9 URL: ${{ secrets.CONTRAST__API__URL }}
10 run: |
11 contrast-cli --scan ./target/${{ inputs.SERVICE_NAME }}-${{ steps.build-
```

```
service.outputs.version }}.jar \
12 --api_key $CT_API_KEY \
13 --authorization $CT_AUTH_TOKEN \
14 --organization_id $ORG_ID \
15 --host $URL \
16 --project_name MY-Project \
17 --language JAVA --wait_for_scan
```

Example: Scan integration with Jenkins

Review the Scan integration steps (page 480) before you integrate Scan with Jenkins.

Contrast Security can share these scripts to integrate scans with a Jenkins pipeline (contact 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.
- **Jenkins pipeline script for use with a Docker container:** The `Jenkins_Script_SCAN_GH_Container_Image` script uses a Docker container (page 463) with the pipeline script.

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 (for example, the Docker plugin)
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`, `CONTRAST_PAT`
   - `CONTRAST_PAT` is the Personal Access Token that Contrast Support gives you. It provides the access to log in to GitHub and download and the Scan local engine file.
   - 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.

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 29). 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 197)
- .NET Framework configuration (page 140)
- Go configuration (page 415)
- Java configuration (page 70)
- Node.js configuration (page 239)
- Python configuration (page 298)
- Ruby configuration (page 357)

Contrast options

The Contrast web interface provides additional options for server configuration:

- Configure a server (page 484)
- Output data to syslog (page 486)

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 749) 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 33)for Java, .NET Framework, .NET Core, Ruby, and Python.

   - Select **Enable sampling for higher performance**.
     This setting is available when Assess is enabled.
     With sampling, Contrast selectively analyzes requests to avoid repeat analysis:
     1. 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.
     2. Afterwards, if Contrast continues to see the same URL, it only checks it 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..

     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 486)
  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.

Send output to syslog

Contrast allows you to send security logs to a remote syslog server in addition to the Contrast Security log. Syslog data is in common event format (CEF) and can be parsed by most security incident event management (SIEM) software.

IMPORTANT

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.

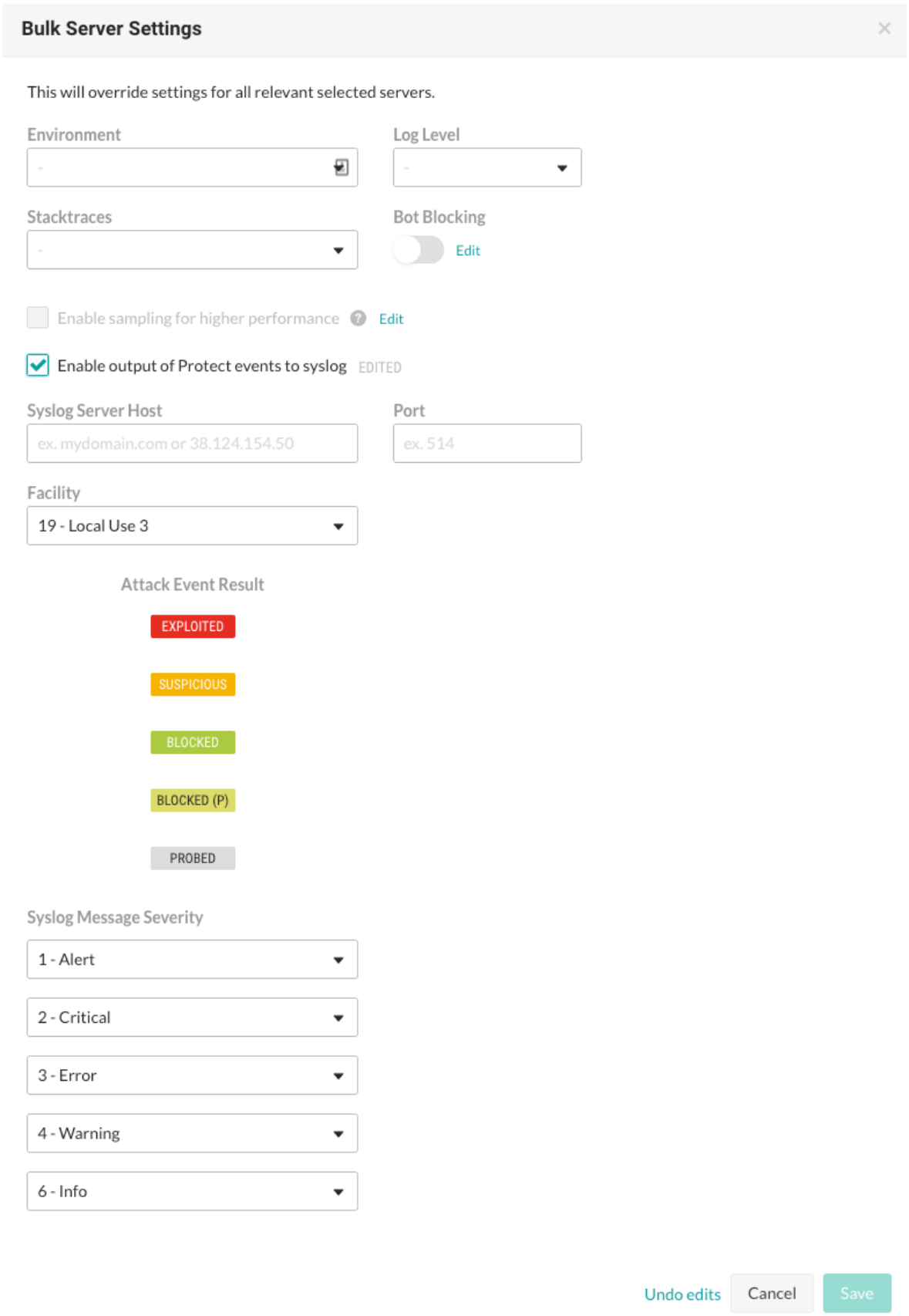
1. When configuring the default organization server settings (page 640), select the checkbox to **Enable output of Protect events to syslog**, which reveals additional fields, and then enter the appropriate settings.
2. Select **Servers** in the header to enable and configure syslog output to an individual server or multiple servers at one time. If syslog defaults have already been set at an organization level (page 640), 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.



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.

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 496) (A-F), public libraries are open-source libraries sourced from Maven (Java), NuGet (.NET), npm (Node.js), RubyGems (Ruby), PyPI (Python), pkg.go (Go), 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 also confirms if the library is used at runtime.

To do this, Contrast creates a hash of the library file, which is used to compare the file's content to a database of known library files. If the hash is in the database, Contrast is able to assign a score (page 496) to the library, provide library version information and report on the total vulnerabilities (CVE's) that have been 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 696).

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.commonarchivecore.ignore.web.fragment` to "true".

Also, any `wsadmin` calls must have the same parameter:

```
wsadmin -javaoption "-
Dorg.eclipse.jst.j2ee.commonarchivecore.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.

- View libraries (page 489)
- View open-source licenses (page 495)
- Learn about library scoring (page 496)

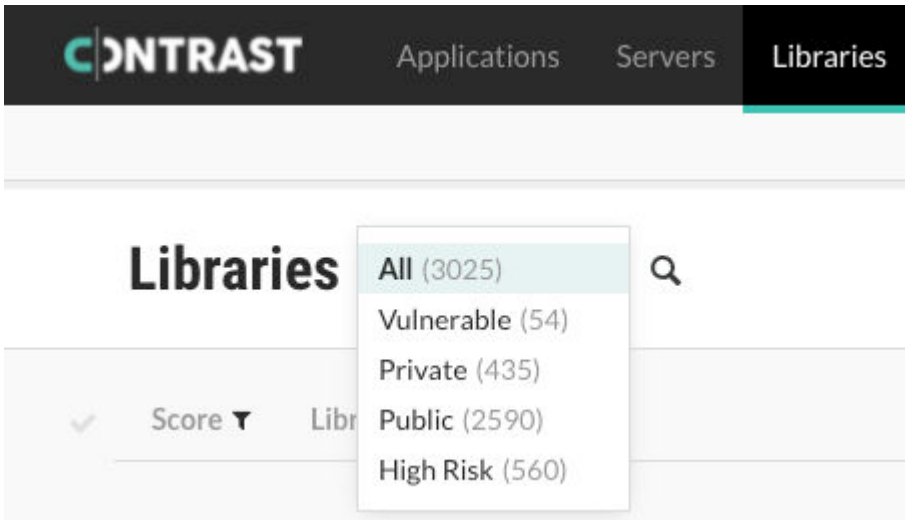
See also

CVE search (page 497)

View libraries

There are multiple ways to view 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.
- Select the small triangle at the very top of the libraries grid to filter the libraries view. You can also click on the magnifying glass icon to search for specific libraries.



The filters include:

- **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 496) of C or below.

- You can also use the column headers with filters in the grid to filter by score, library, and application. The libraries grid shows:
  - **Score:** Shown as a letter grade using this scoring guide (page 496).
  - **Library:** Click a library name in the grid to go to its details panel. This is where any known vulnerabilities (CVEs) that Contrast has found within the library will be listed along with a list of the applications and servers where the library appears.

    Use the filters to narrow the results:
    - **Languages:** Locate vulnerable libraries by a specific language
    - **Licenses:** View libraries by licensed applications
    - **Environments:** Helps to easily locate any vulnerable libraries in production
    - **Servers:** Find vulnerable libraries by server type
  - **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:** 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 in a list and are color-coded by severity. Vulnerabilities with the critical severity status appear at the top of the list and are coded red.

  - **Application:** Lists applications using the library.
  - **Usage:** 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 increases. Click the number to analyze the library usage (page 492). There you can see information on classes loaded as well as the risks and policy violations associated with the library.
  - **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 vulnerabilities that are acknowledged and the risks are acceptable or the library is not used.
    - ***Remediated*:** The vulnerable library has been remediated.
    - ***Reported*:** When a library with vulnerabilities is detected by Contrast.
- 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 496) that is grade C or below.

Discover or delete libraries

Libraries are associated to 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 640)).

Libraries can also be manually deleted from an application.

To delete libraries:

1. Select **Libraries** in the header.
2. Hover over the grid row with the library 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 library details page.

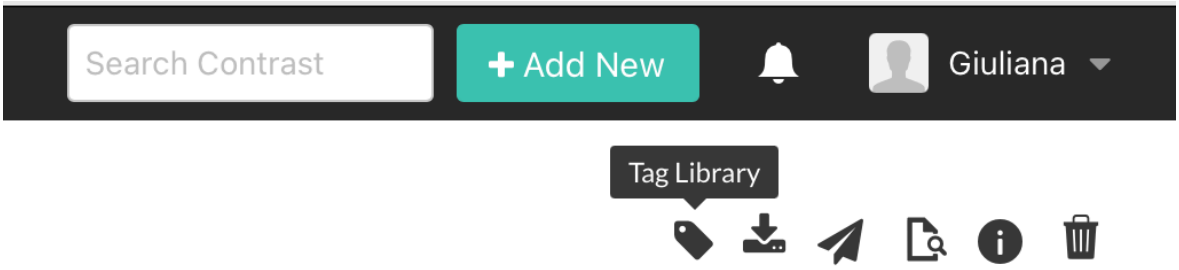
To delete multiple libraries at once, use the check marks in the left column to select the libraries you want to delete, then select the **Delete** icon from the batch action bar that appears at the bottom of the page.

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.

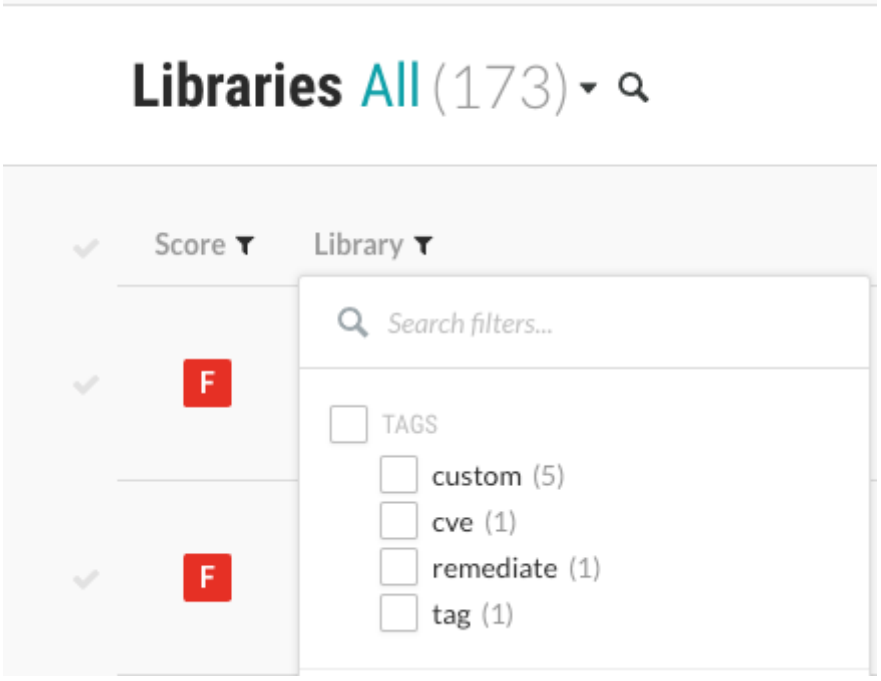
Add tags to libraries

To add tags to libraries:

1. Select **Libraries** in the header, then hover over the row in the grid for the library you want to tag.
2. In the far right column, select the **Tag** icon. 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 filter next to the **Library** column of the grid, then select the tags to filter.



6. You can also see tags next to the library name on the library's details page, and remove them by selecting the **X**.

Track libraries with bugtrackers or email

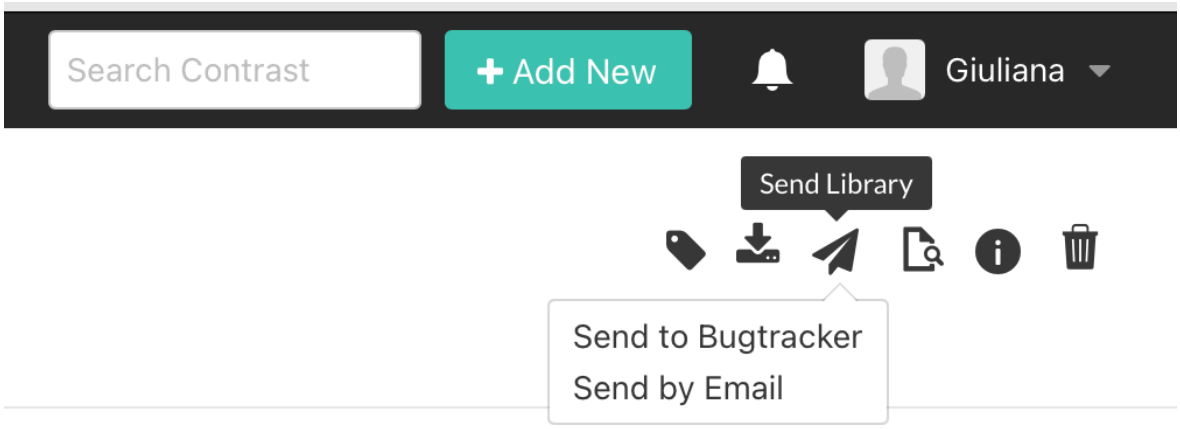
Track vulnerable libraries by sending library details to your email address or to an integrated bugtracker service that creates tickets for your developers.

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 496)

To receive this information for a particular library:

1. Select **Libraries** in the header, then hover over the row in the grid for the library you want to track.
2. In the far right column, click the **Send** icon 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.



3. In the window that appears, select the integrations (page 553) you want to use. Select the **Issue type**, **Default Priority**, **Default Assignee**, and **Reporter**, then select **Send** to create the ticket. For email, enter the email address then select **Send** to send the email.
4. 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.

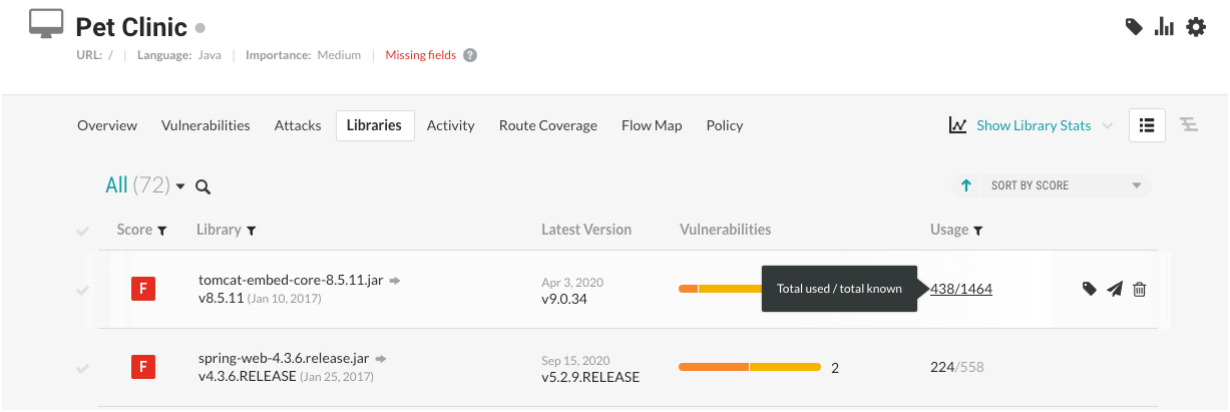
Analyze runtime library usage

Runtime library usage gives insight into which parts of a library are actually 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 22) can see full usage details. To learn more, contact our sales department at **sales@contrastsecurity.com**.

Select **Libraries** in the header and view the **Usage** column to see if a library is used at runtime, and how much. 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.

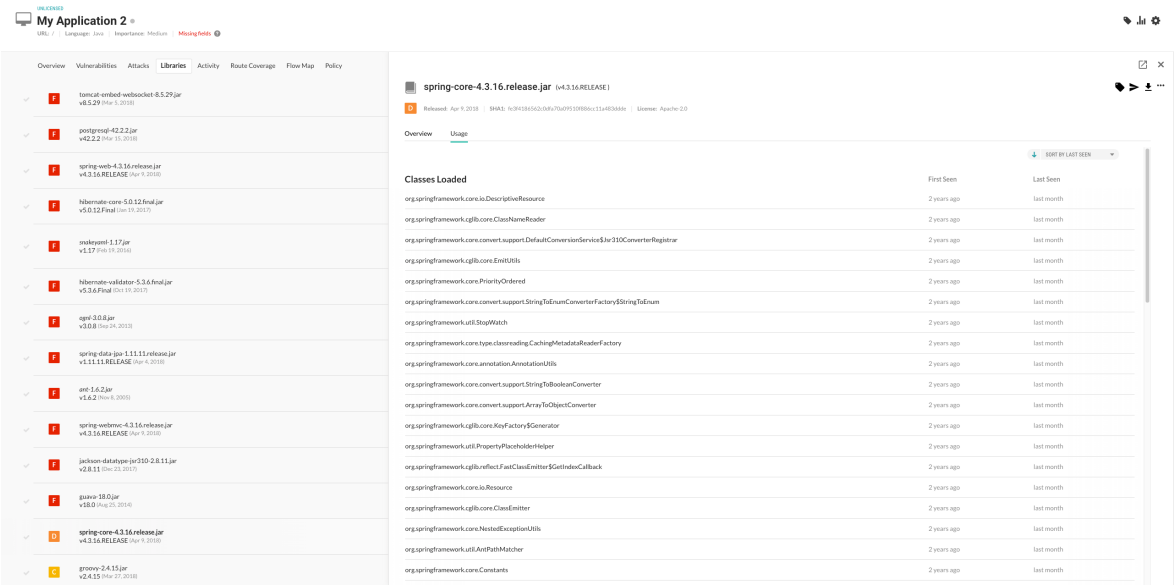


Items loaded may be classes, files, or functions, depending on the languages of the applications using this library. If a primary application contains multiple applications that use the same library, the classes loaded will be representative of each merge application.

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.



4. 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 494) 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.

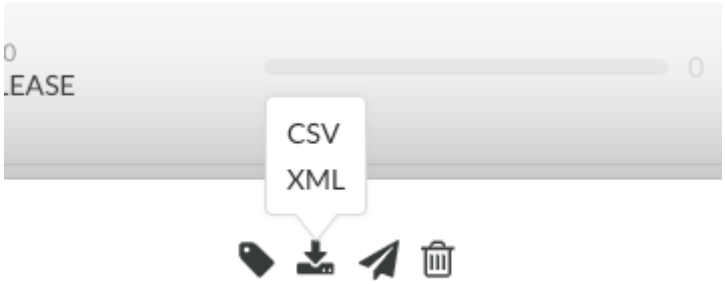
5. You can also tag (page 491) and track (page 492) the library.
6. Click the **More** menu (⋯) to view package details, repository location, delete (page 490) the library, or export usage details in CSV format.
7. Select the **X** to close the details panel.

Export library details

The exported file contains the following data fields for each library:

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.
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).



3. The export will download to your desktop. Exports contain the following data fields for each library:
   - Library Name
   - Language
   - Version
   - Release Date
   - Latest Version
   - Score
   - SHA1
   - CVE Count
   - Application Count
   - Server Count
   - Number of Classes
   - Number of Used Classes
   - Open source License
   - Policy Violation

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 494).

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 511) 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 511) 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 511).

- 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 442) 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 646) 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 x 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 x 10 = 20
```

If you have a library with the scores 2.4 and 2.2, the penalty would be:

```
2.4 x 10 = 24
```

The final score of the library is calculated by subtracting each of the three penalty values from 100.

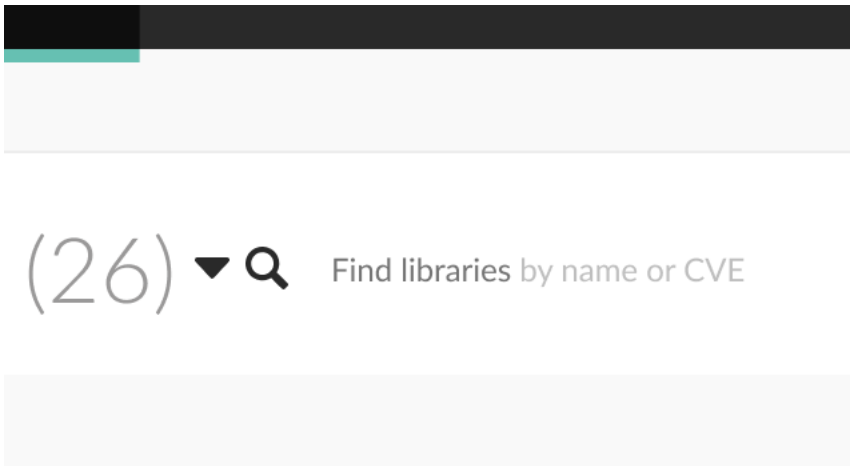
```
100 - 5 - 20 - 24 = 51
```

A score of 51 maps to a letter grade of F.

CVE search

Use the search bar to locate specific CVEs in libraries (page 488).

- 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 626), Application (page 438), and Server (page 483) libraries.

Serverless

Contrast Serverless Application Security features dynamic scanning, static scanning, graph visualization, and resource observability that help you maintain awareness of your environments.

With Contrast Serverless Application Security you can:

- Scan functions on demand (page 500)
- View results (page 501)
- Change inventory criteria (page 506)
- Change Serverless scan settings (page 507)
- View function and service relationships (page 507)

See also

- Integrate Contrast Serverless with JIRA (page 591)

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 506)

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.

Supported languages for Contrast Serverless

We support the following programming languages for Contrast Serverless Application Security.

- Java
- Node.js
- Python

Supported platform for Contrast Serverless

We support the following platform for Contrast Serverless Application Security.

- AWS Lambda

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 prior to deployment and recommends permission remediations.
  - Contrast SCA - Provides SCA for open-source libraries using the Contrast SCA engine.

  During the scan, Contrast temporarily instruments the function with a layer and configuration changes. Once the scan is complete, Contrast restores the function to its original settings.
  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.
  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 OWASP Top 10 benchmark. For example:

- SQL injection
- Code injection
- Local file inclusion

  During a dynamic scan, Contrast tries to send malicious input to the code and then, exercises the code to discover vulnerabilities.

  This action has no effect on your code, however, a scanned function is invoked.
- **Continuous monitoring**

  Once you connect to an AWS account from Contrast, Contrast Serverless Application Security monitors this account. As you make changes to your functions' code or configurations, Contrast automatically initiates a new scan.

Get started with Contrast Serverless

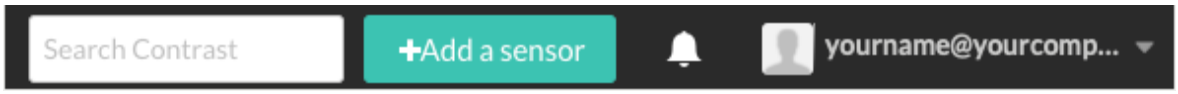
To start using Contrast Serverless, open Contrast and connect to your AWS account to create a new stack.

Before you begin

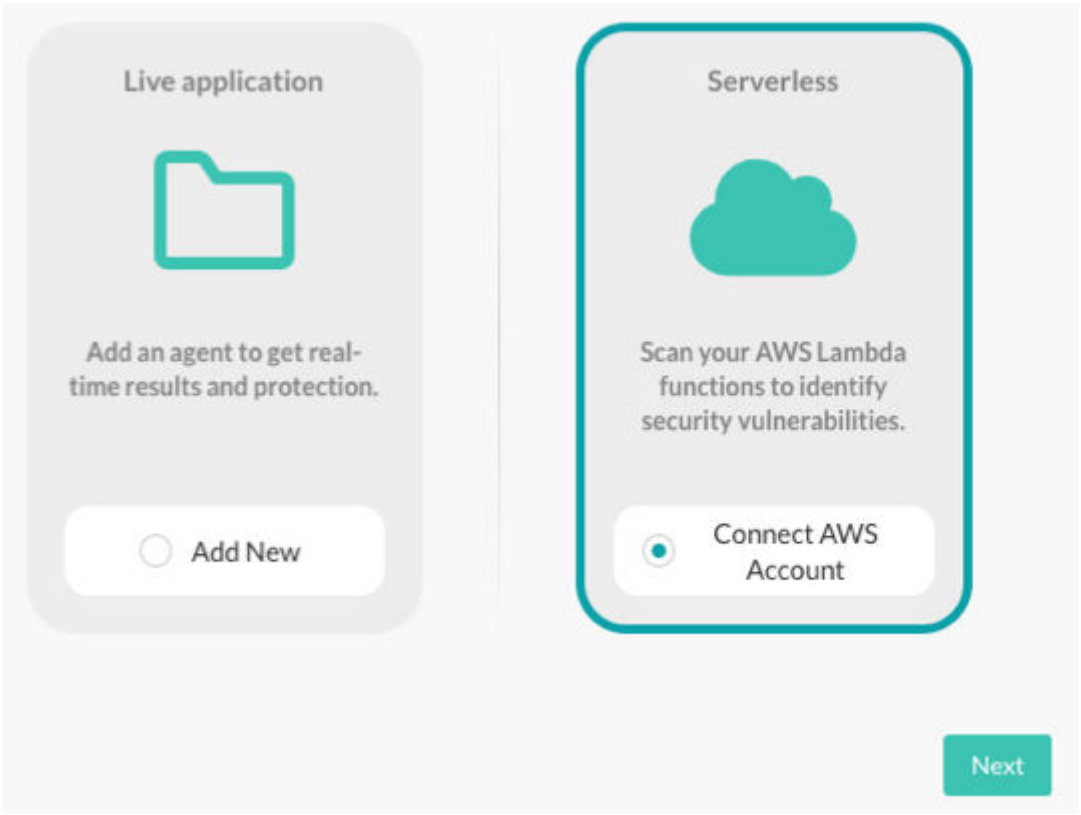
- Have your AWS account information available.
- Permissions to deploy a stack in an AWS account are required.

Steps

1. In the Contrast application, select **Add New** at the top of the page.



2. Select **Connect AWS account** and click **Next**.



3. (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.

You can change these settings (page 507) at any time in the Settings tab.

4. Select **Create new stack**.
5. 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.
6. Approve the stack deployment in your account.
   The stack deployment takes approximately two minutes to complete.
7. Return to the Contrast application and verify that the Account connected and Scan started messages are displayed.
8. 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 500)
- View results (page 501)
- Change inventory criteria (page 506)
- Change scan settings (page 507)

Scan functions on demand

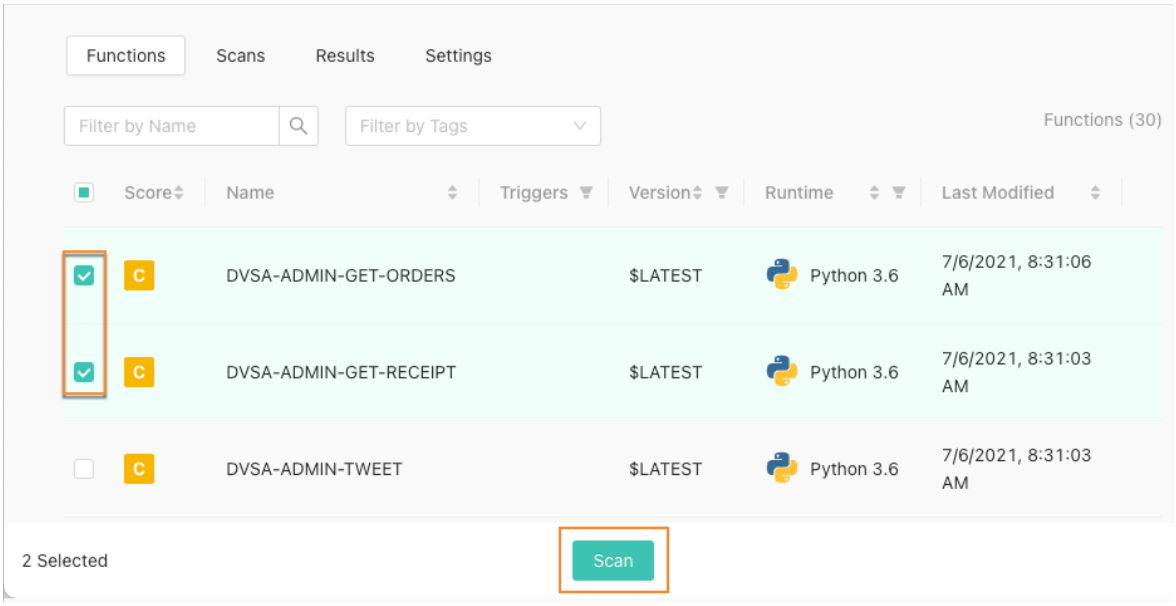
Although scans of all functions in your AWS account occur automatically, you can also scan a specific function on demand.

Before you begin

- Identify the functions that you want to scan.

Steps

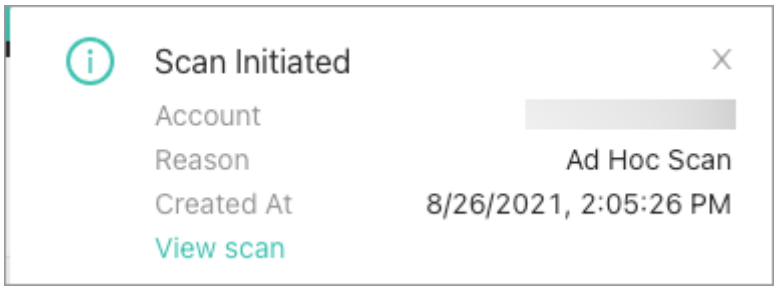
1. Select **Serverless** in the header.
2. Select the check box next to one or more functions that you want to scan.



3. Select **Scan**.
4. In the Confirm Scan window, verify or change the scan type settings for the selected functions and select **OK**.

- **Scan for Least Privilege violations and CVEs:** Required for static scanning.
- **Scan for custom code vulnerabilities:** Required for dynamic scanning.

The Scan initiated message displays.



5. 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.

View results

View results to see details about vulnerabilities for permissions, dependencies, exploits, and CVEs.

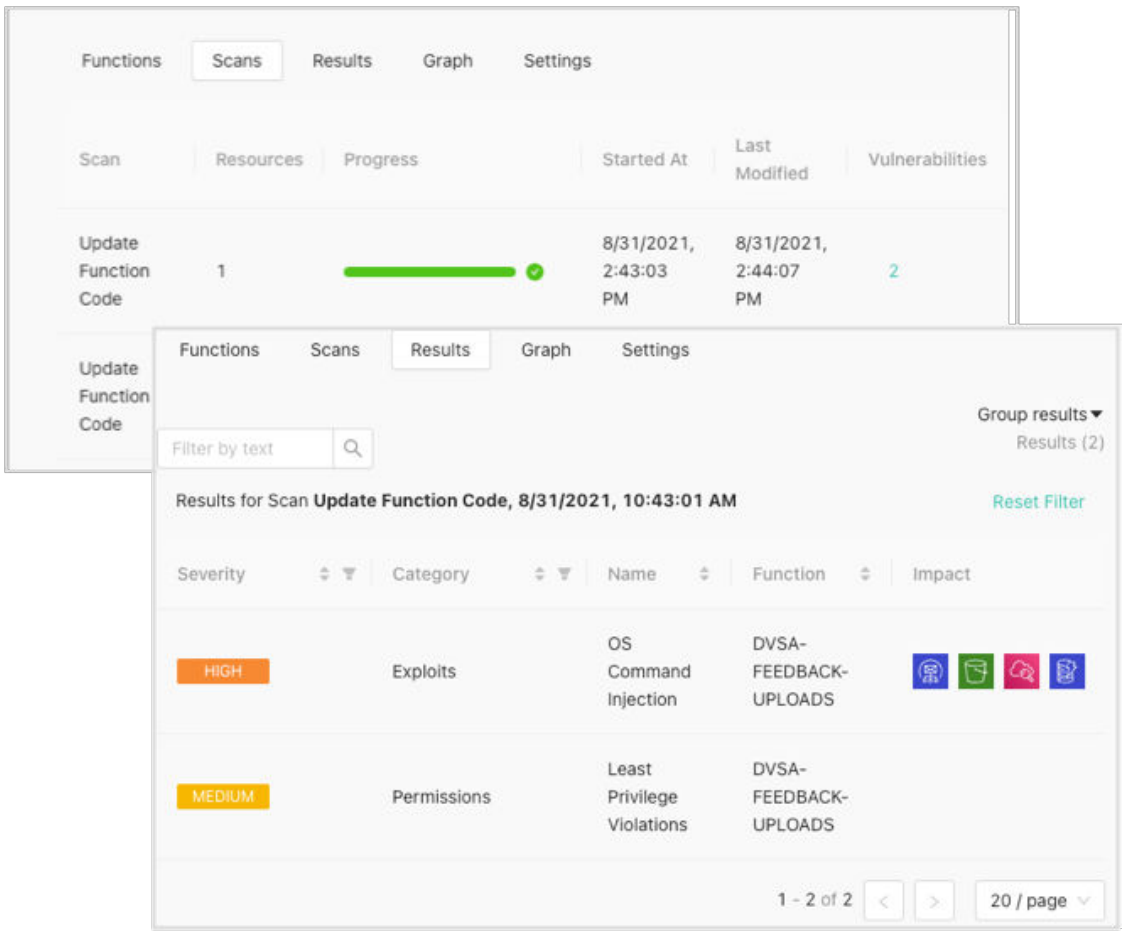
Before you begin

- Ensure that at least one scan is complete.

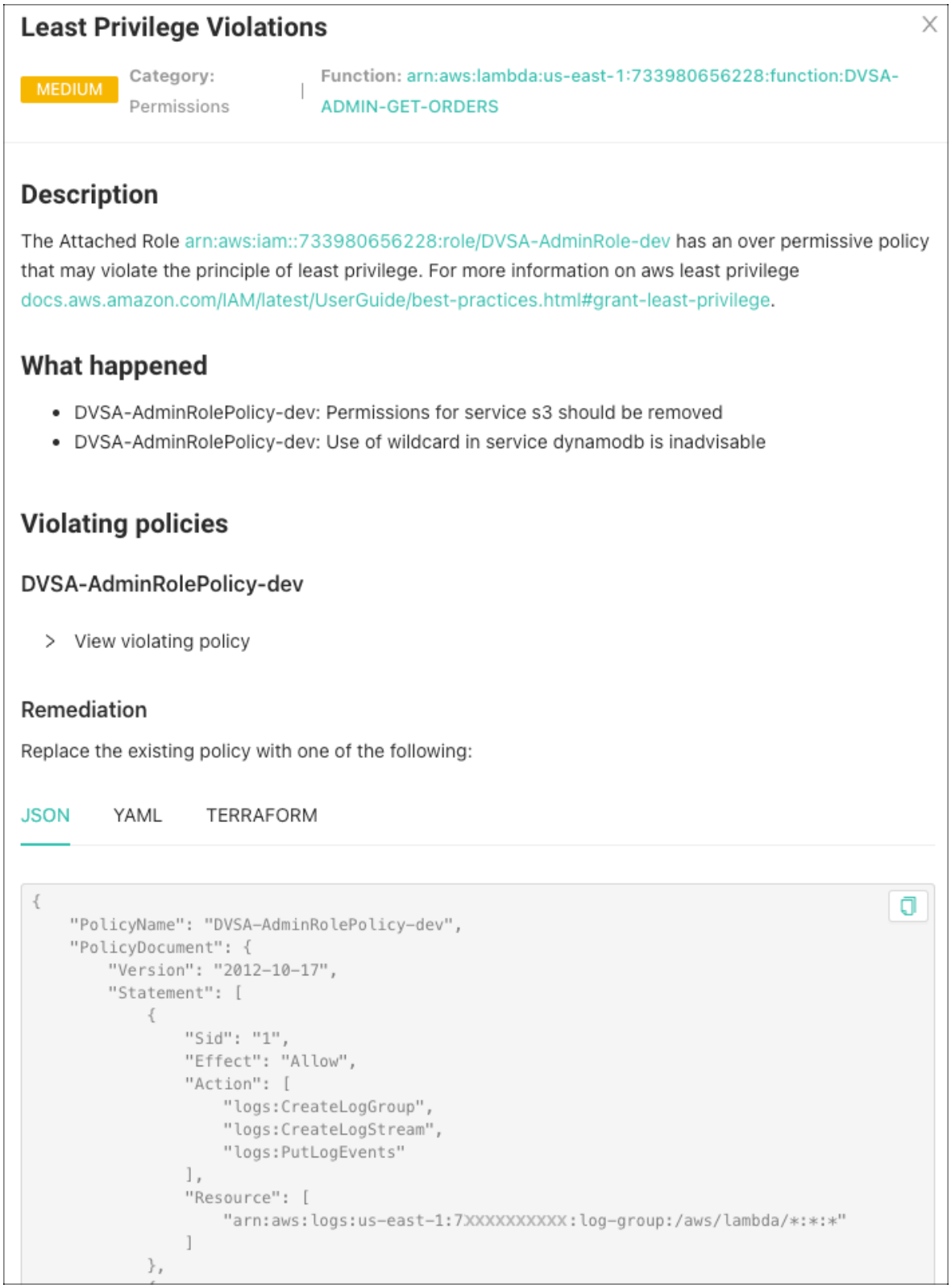
Steps

1. Select **Serverless** in the header.
2. To view results for all scans, select **Results**. See Scan status details (page 504) 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 747) for information about the levels.
   b. The **Category** is based on vulnerabilities in the type of function.
   c. Results can also be grouped by **Category** or **Function** by selecting the option under **Group results** on the right-hand side of the screen.

   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, select the **Magnifying glass** icon ( )
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 a scan details page, select a number in the Vulnerabilities column for a function.



The results detail page looks similar to this example:



Result details

The results details that you see 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.

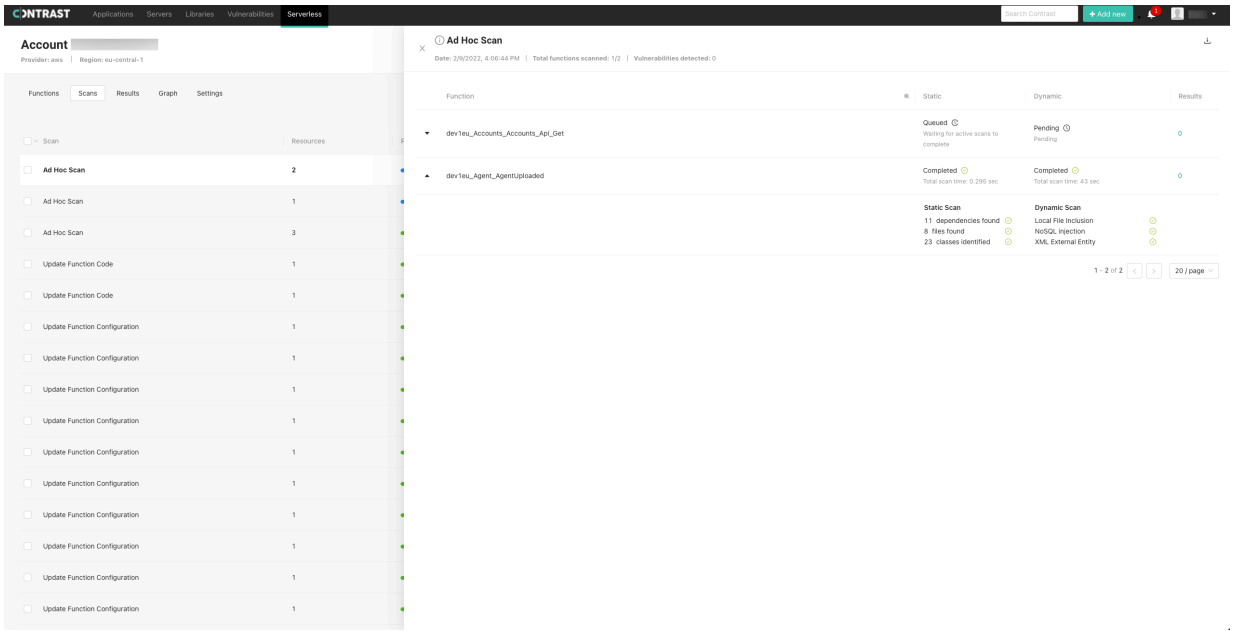
You might also see these details:

- **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 Settings (page 507) To scan, change the Inventory settings or run an ad-hoc scan on the function.
Canceled	Newer scan initiated A newer version of the function is already in queue.
	Lambda state inactive

Status	Description
	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 501) tab.

Download serverless scan results

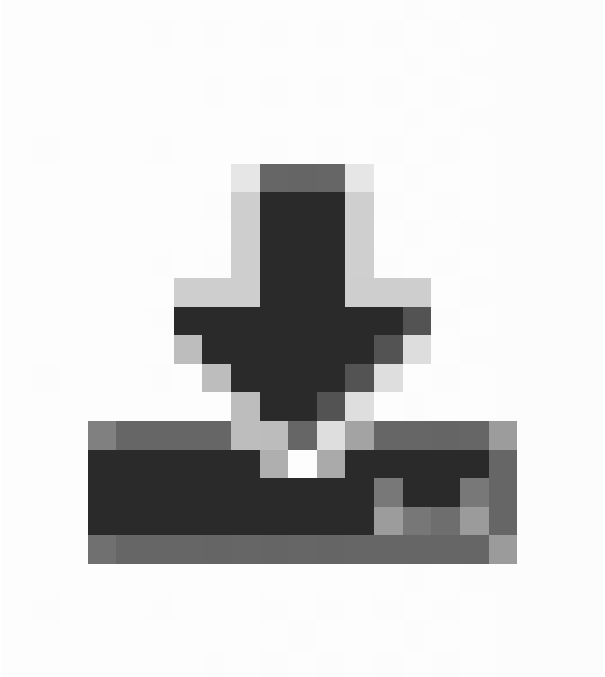
After a scan completes, you can download the results as a CSV file.

Before you begin

- Identify the scan with results you want to download.

Steps

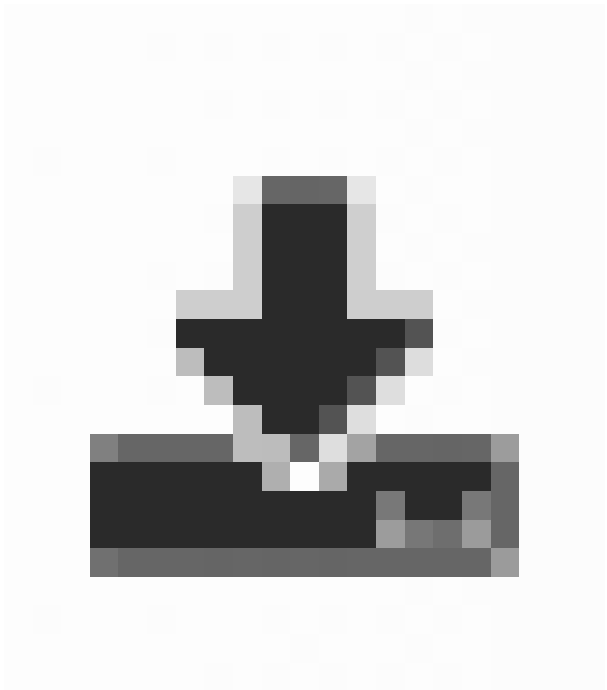
1. Select **Serverless** in the header.
2. Select either the **Scans** tab or the **Results** tab.
3. To download scan results from the **Scans** tab, hover over the end of a row for a scan and select the download icon.
4. To download scan results from the **Results** tab:
   a. Hover over the end of 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



( ) on the right-hand side.

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.

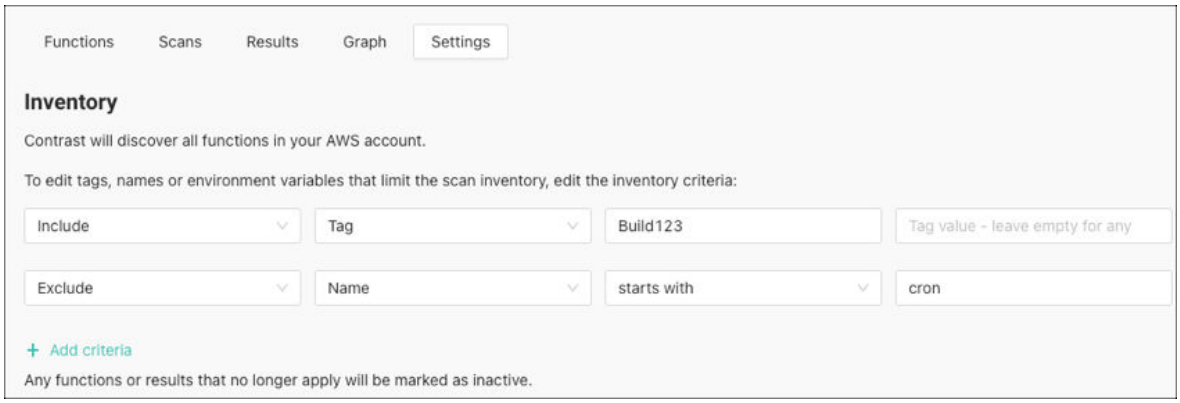
Excluding a function excludes it from the inventory and scan results.

Before you begin

- Identify the functions you want to include or exclude in scans.
- An Admin role is required to change the inventory criteria.

Steps

1. Select **Serverless** in the header.
2. Select the **Settings** tab.
3. 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**.



4. 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 500) for a manual scan of selected functions.

Before you begin

- Determine if you want to use static scans, dynamic scans, or both.
- An Admin role is required.

Steps

1. Select **Serverless** in the header.
2. Select the **Settings** tab.
3. Under Scan, select the types of scans that you want to use:
   - **Scan for Least Privilege violations and CVEs:** Use this setting for a static scan.
     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.
   - **Scan for custom code vulnerabilities:** Use this setting for a dynamic scan.
     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.

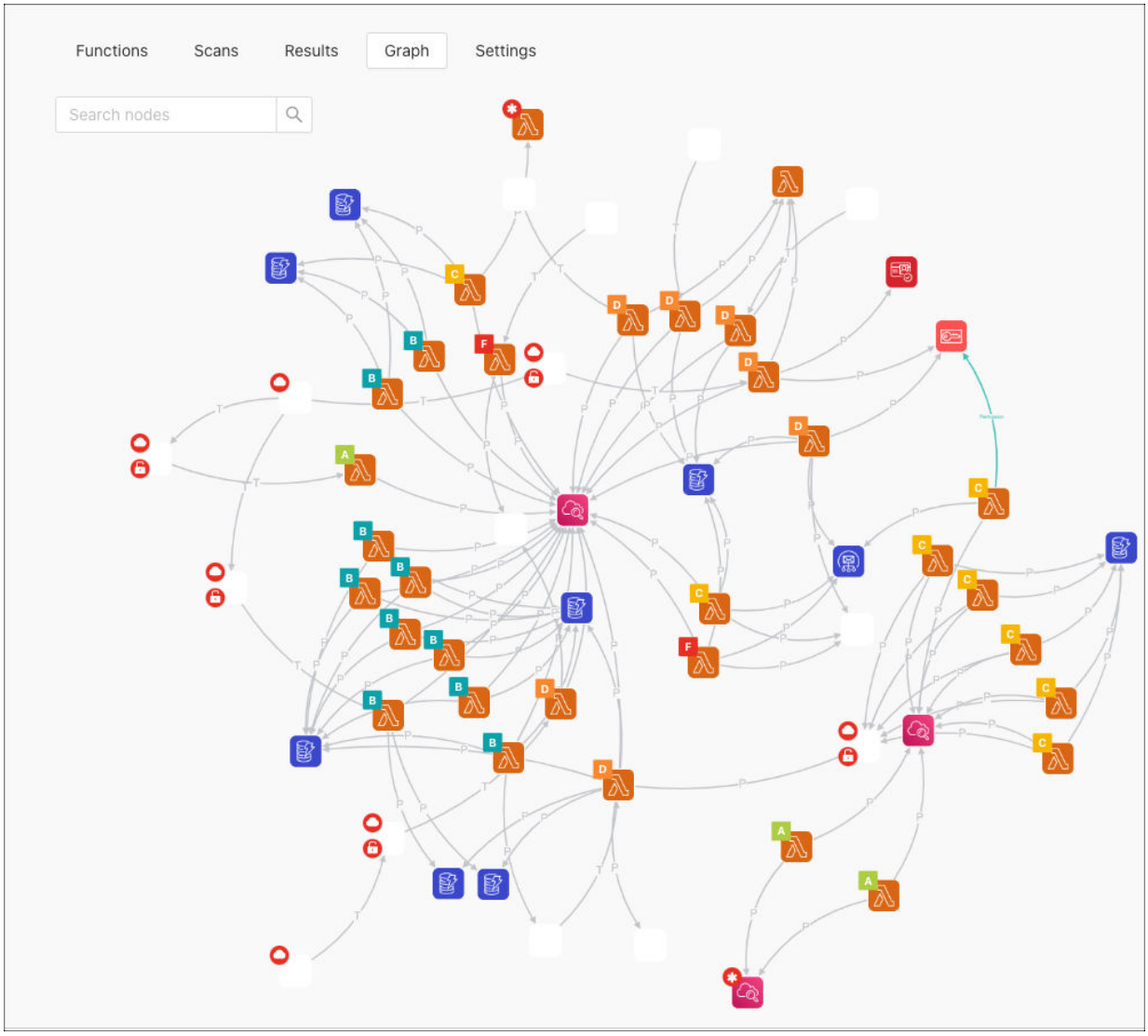
   IMPORTANT
   Serverless scans do not change your function code.
4. Select **Save.**

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:

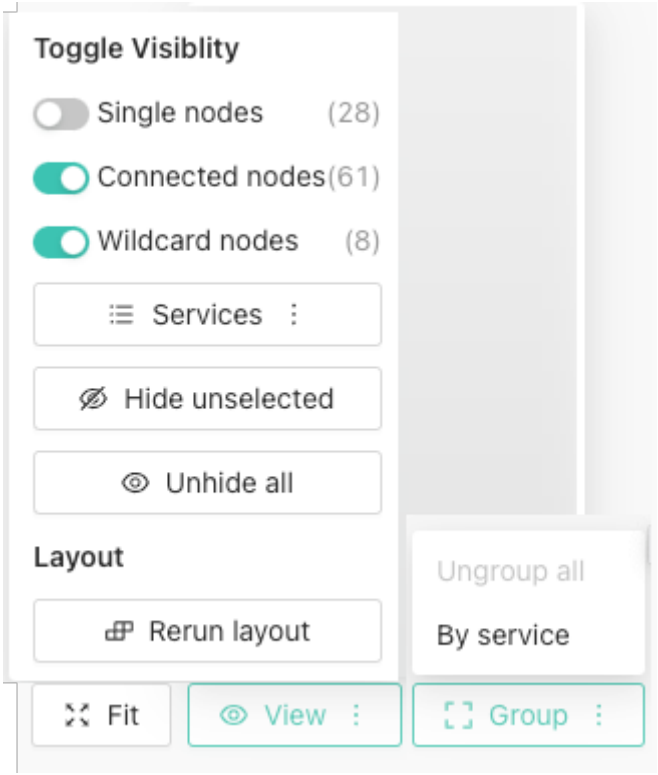
- AWS tags
- Vulnerability results

- 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.



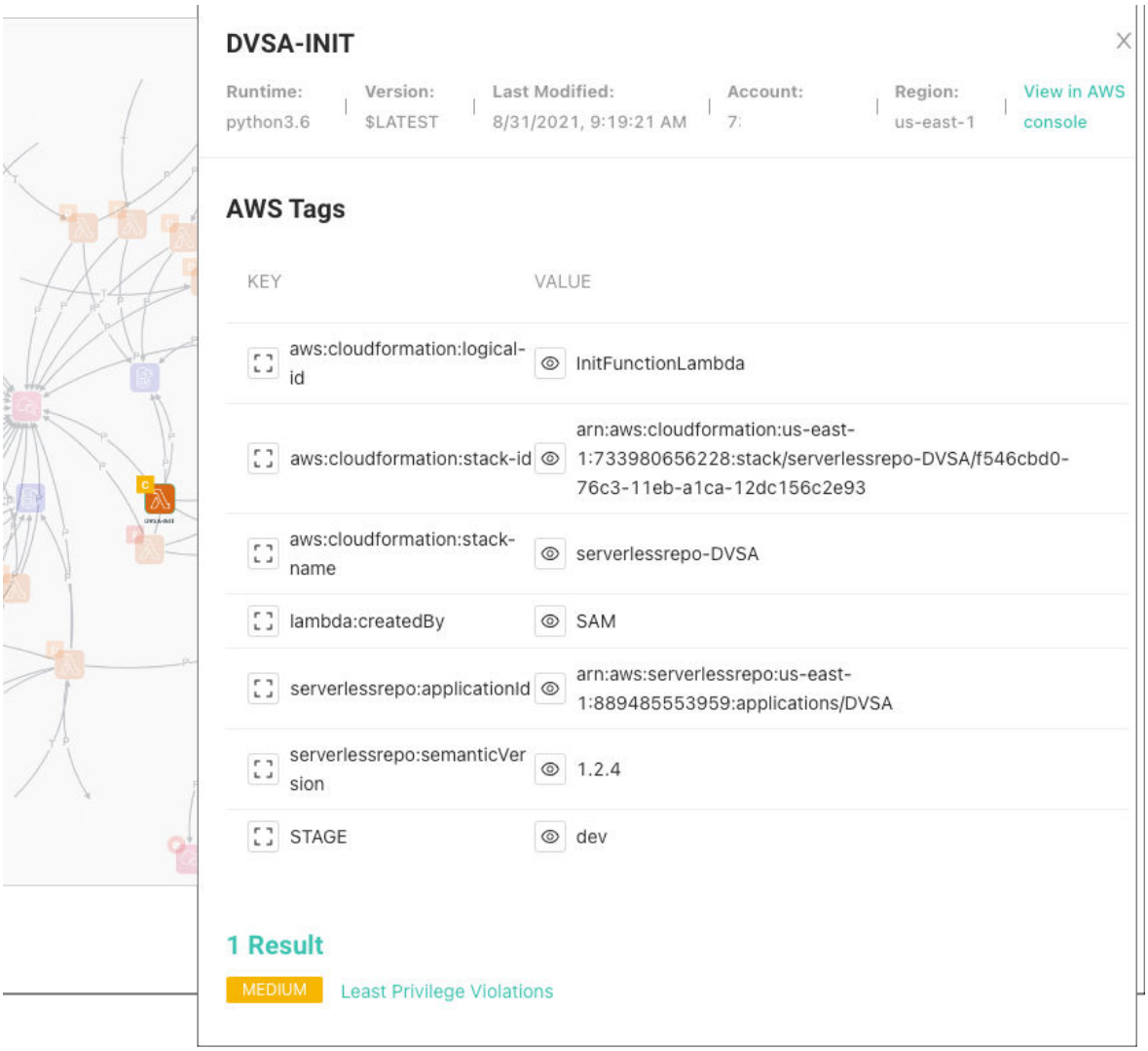
Steps

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 AWS tags to filter the view.



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.

Uninstall Contrast Serverless

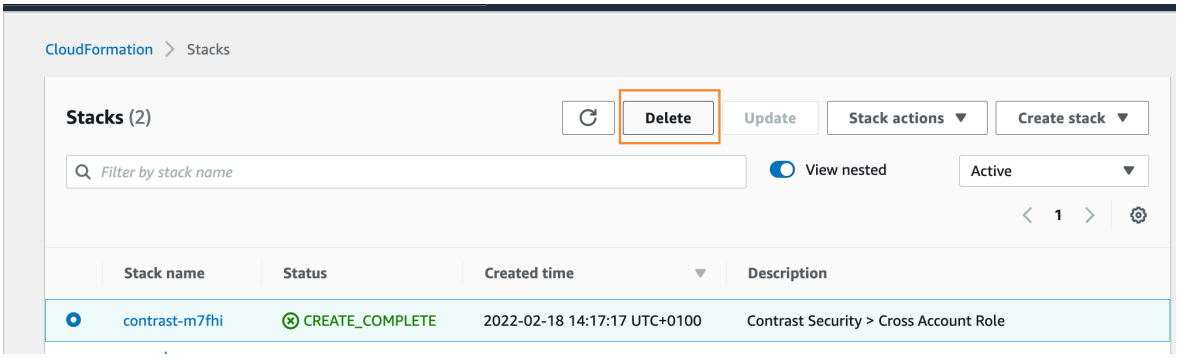
Uninstalling Contrast Serverless requires the deletion of a stack in the AWS console.

Before you begin

- Identify the objects that you want to remove

Steps

1. Log in to your AWS account or use the AWS CLI/API.
2. 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. Use the CLI locally or automated in your CI/CD pipeline.

Before you start

A few things to keep in mind:

- CodeSec users start here.
- Enterprise Contrast users can install (page 511) 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

Get started scanning with easy-to-follow steps. If you already have a Contrast account, you can start scanning using these commands (page 512).

See also

- Install Contrast CLI (page 511)
- Run Contrast CLI (page 512)

Install Contrast CLI

Install Contrast CLI before running a scan.

With Homebrew

Install from Contrast's tap with Homebrew by running the following commands.

```
brew tap contrastsecurity/tap
brew install contrast
```

With NPM/YARN

Install using npm or yarn `@contrast/contrast`.

```
npm install --location=global @contrast/contrast
```

With Binaries

- Go to artifacts.
- Download the latest package.
- You must allow *execute* permissions on the file depending on your OS.

Once installed, continue by running the Contrast CLI (page 512).

Run Contrast CLI

If you are an Enterprise Contrast user, you can use the new Contrast CLI to run a SAST scan or find vulnerable libraries and then view results in Contrast.

Before you begin

- You must be a current Contrast user.
- You must have the Contrast CLI installed (page 511).
- In Contrast, under **user menu** > **User settings** > **Profile**, locate and copy this information:
  - API key
  - Organization ID
  - Authorization header
  - Contrast URL (host domain). For example, https://app.contrastsecurity.com.

Steps

- **Run a SAST scan:**

1. In the terminal, type the following code: `contrast scan -f <file name>`

NOTE

If you have not used the `auth (page 513)` command to store credentials locally, use the following set:

```
contrast scan -f file name>
--organization-id <your organization ID>
--api-key <your API key>
--authorization <your authorization header>
--host <your host domain>
```

2. In the results click the link to view the scan results (page 468) in Contrast.

- **Find vulnerable libraries:**

1. In the terminal, type the following code: `contrast audit`

NOTE

If you have not used the auth (page 513) command to store credentials locally, use the following set:

```
contrast audit
--organization-id <your organization ID>
--api-key <your API key>
--authorization <your authorization header>
--application-id <your application ID>
--host <your host domain>
```

2. In the results click the link to open the dependency tree view (page 495) in Contrast.

Contrast CLI commands

The following is a listing of the commands available to basic and advanced users of Contrast CLI.

Usage: `contrast [command] [options]`

audit

Searches for a dependency configuration file in the working directory to perform a security audit of dependencies and returns the results.

Java	*pom.xml* and Maven build platform including the dependency plugin *OR* *build.gradle file* and gradle dependencies or ./gradlew dependencies must be supported
.NET Core	MSBuild 15.0 or greater and a packages.lock.json file. Note: If the packages.lock.json file is unavailable it can be generated by setting `RestorePackagesWithLockFile` to true within each *.csproj file and running dotnet build.
Node	*package.json* and a lock file (either *package-lock.json* or *yarn.lock*)
PHP	*composer.json* and *composer.lock* files
Python	*pipfile* and *pipfile.lock* files
Ruby	*gemfile* and *gemfile.lock* files
Go	*go.mod* file

- **Usage:** `contrast audit [option]`

NOTE

By default, results are not held or stored, which would allow you to do local checks via your console. Add the `--track` flag to the `contrast audit` command to view your SCA results via the dependency tree (page 495) in the Contrast web interface.

- **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`

- `--ignore-dev`
  Excludes developer dependencies from the results. All dependencies are included by default.
  Alias: `-i`
- `--save`
  Generate and save an SBOM (Software Bill of Materials). Valid options are: **--save spdx** and **--save cyclonedx** (CycloneDX is the default format.).
  Alias: `-s`
- `--severity`
  Specify the 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*.

- **Advanced options:**
  - `--api-key`
    Required for Enterprise users. Agent API key provided by Contrast. See agent keys (page 32)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 32) to find the ID.
  - `--tags`
    Apply labels to an application. Labels must be formatted as a comma-delimited list. For example, label1,label2,label3.
  - `--track`
    Send your dependency audit to Contrast to see the results in the Contrast web interface and start automating security checks.
- **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>".

auth

For CodeSec users, authenticate using your GitHub or Google account. A new browser window will open for log in.

- **Usage:** `contrast auth`

If you already have a Contrast account, run the following `auth` command to store your credentials locally.

- **Usage:**

```
contrast auth
--api-key <your API key>
--authorization <your authorization header>
--host <your host domain>
--organization-id <your organization ID>
```

You can then start scanning using these commands (page 512).

config

Displays stored credentials.

- **Usage:** `contrast config`
- **Options:**
  - `-c`, `--clear`
    Removes stored credentials.

help

Displays usage guide. To list detailed help for any CLI command, add the **-h** or **--help** flag to the command.

- **Usage:** `contrast scan --help`
- **Alias:** `-h`

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`
  - `--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>".

learn

Launch Contrast's Secure Code Learning Hub.

- **Usage:** `contrast learn`

scan

Performs a security SAST scan.

- **Usage:** `contrast scan [option]`
- **Options:**
  - `--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`
  - `--host`
  Required for Enterprise users. The host name. For example, https://app.contrastsecurity.com/.
  - `--language`
  Valid values are JAVA, JAVASCRIPT, and DOTNET.
  Alias: `-l`
  - `--name`
  Contrast project name. If not specified, Contrast uses *contrast.settings* to identify the project or creates a project.
  Alias: `-n`
  - `--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`
  Specify the severity of a vulnerability to fail a build. Use with the **--fail** flag. For example, **contrast scan --fail --severity high**. Severity levels are *critical*, *high*, *medium*, *low*, or *note*.
- `--timeout`
  Time in seconds to wait for the scan to complete. Default value is 300 seconds.
  Alias: `-t`

- **Advanced options:**
  - `--api-key`
    Required for Enterprise users. Agent API key provided by Contrast. See agent keys (page 32)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 32) 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>".

version

Displays Contrast CLI version.

- **Usage:** `contrast version`

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 511).

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 518) page.

With this composition analysis you can:

- Identify vulnerable libraries
- Fail a build based on CVE severity
- View a dependency tree (page 495) 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 519) so you can register new applications (page 519)and begin analyzing your libraries during the development phase using the command line options. (page 521)

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 `pom.xml` file, and have the Apache Maven Dependency plugin. To test if the CLI works with your project, build a dependency tree by running `mvn dependency:tree`.
	Gradle (v4.8 or above)	A *build.gradle* file is required and **gradle dependencies** or **./gradlew dependencies** must also be supported.
.NET Framework and .NET Core	MSBuild 15.0 or greater and a *packages.lock.json* file	If the *packages.lock.json* file is not present, it can be generated by setting `RestorePackagesWithLockFile` to **true** within each *csproj* and running .NET build.
Node.js	You must have either a *package-lock.json* or a *yarn.lock* file present.	Vulnerability reporting is supported for front-end technologies like React or Angular.
PHP	You must have the *compose.lock* and the *composer.json* files present.	
Python	The *pipfile* and *pipfile.lock* files.	
Ruby	The *gemfile* and *gemfile.lock* files.	
Go	The *go.mod* file.	

NOTE
Only single language applications are supported at this time.

Legacy Contrast CLI - Install

To install the Contrast CLI (page 518):

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 519) that has not yet been instrumented. However, all applications should be instrumented so that your application has a library score (page 496) 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 518).
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 519) to begin analyzing your code.

Legacy Contrast CLI - Register applications

Once you install the Contrast CLI (page 519) 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 436). 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 495).

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 30) so that the library score (page 496) 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 521).

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
`--api_key <string>`	An agent API key (page 32) provided by Contrast. (required)
`--application_id <string>`	The ID of the application cataloged by Contrast. (required)
`--application_name <string>`	The name of the application cataloged by Contrast. (optional)

Command	Description
`--authorization <string>`	User authorization credentials provided by Contrast. (required)
`-h`, `--help`	Displays the help guide.
`--host <string>`	The name of the host and, optionally, the port expressed as `<host>:<port>`. Does not include the protocol section of the URL (*https://*). Defaults to *app.contrastsecurity.com*. (optional)
`--language <string>`	Valid values are JAVA, DOTNET, NODE, PHP, PYTHON, RUBY, and GO. If there are multiple project configuration files in the `project_path`, language is required. (required for catalogue)
`--organization_id <string>`	The ID of your organization (page 32) in Contrast. (required)
`--project_path <string>`	The directory root of a project/application that you want to analyze. Defaults to the current directory. (optional; required if running on Windows).
`--proxy <string>`	Allows for connection over 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>*. (optional)
`--silent`	Silences JSON output. (optional)
`--sub_project <string>`	Specifies the subproject within a Gradle application. (optional)
`-v`, `--version`	Displays the CLI version you are currently using.
`--yaml_path <string>`	The path to display parameters from the YAML file (optional) If `yaml_path` is used, the following connection parameters are ignored from the terminal: • `yamlOnly:` • `key:pathToKey` • `cert:pathToCert` • `cacert:pathToCaCert`

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
Catalog applications	
`--app_groups <string>`	Assigns your application to one or more pre-existing groups when using the `catalogue` command. Group lists should be comma separated. (optional)
`--catalogue_application`	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 <string>`	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 519). (optional)
Register an application	

`--cli_api_key <string>`
`--cli_authorization <string>`
`--cli_organization_id <string>`
`--cli_host <string>`
`--language <string>`
`--application_name <string>`
`--sbom`
	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.
	Note: The "cli_" prefix in the parameters will be deprecated in a future release.

Reports	
`--cve_severity <level>`	Combined with `--report`, allows the user to report libraries with vulnerabilities above a chosen severity level (page 539). 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 549) 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 480).

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
`--project_id <ProjectID>`	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. **Recommended:** For the first scan, use the **--project_name** command instead of this one. Scan creates the project for you.
`--project_name <ProjectName>`	The name of the scan project. If the name includes spaces, enclose it in double quotes ("). 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.
`--save_scan_results`	If provided, will save the SARIF file as *results.json* to the current directory. (optional)
`--scan<FileToBeScanned>`	Starts a static scan of the specified WAR or JAR file. Replace `<FileToBeScanned>` with the path of the WAR or JAR file that you want to upload for scanning.
`--scan_results_file_name`	Must be JSON file format. If provided, will override the default name to save the SARIF file. (optional)
`--scan_timeout`	Set a specific time span (in seconds) before the function times out. The default timeout is 20 seconds if `scan_timeout` is not set.
`--wait_for_scan`	Waits for the result of the scan.

Vulnerabilities

Once you instrument an application (page 30), 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 525)
- View application vulnerabilities (page 526)
- Analyze vulnerabilities (page 533)
- Track vulnerabilities (page 531)
- Fix vulnerabilities (page 533)

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 445).

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** : Custom tags you assigned to vulnerabilities.
     - **Type:** Types of vulnerabilities.
     - **Servers:** Vulnerabilities for applications that the selected servers are hosting.
     - **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
   - **Application:** Available filters are: Application names and custom tags you assigned to 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.

4. 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

See also

View application vulnerabilities (page 526)

View application vulnerabilities

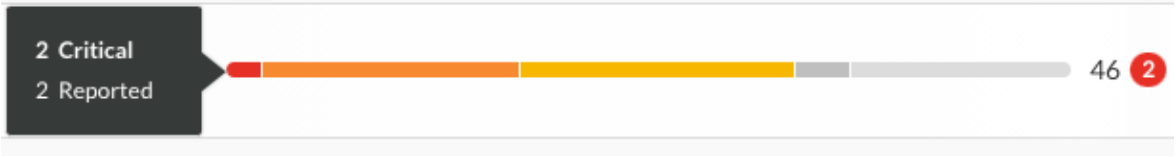
From the Applications list, you can view vulnerabilities for a specific application.

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 445).

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.

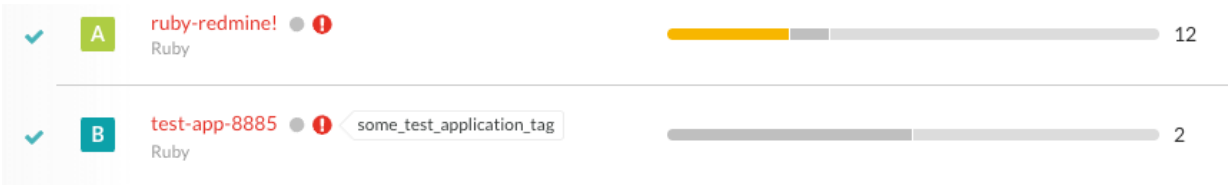
Filtering by sink can help you identify a line of code that is causing multiple vulnerabilities.
  - **URLs:** Vulnerabilities associated with a specific URL.
  - **Compliance policy:** Vulnerabilities associated with selected compliance policies
  - **Routes:** Vulnerabilities associated with selected routes.
- **Application:** Modules included in the application.
  If you are viewing an unmerged application, this filter shows vulnerabilities for the selected application.
  If you are viewing vulnerabilities for a merged application, this filter shows view vulnerabilities for the modules in the merged application.
- **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

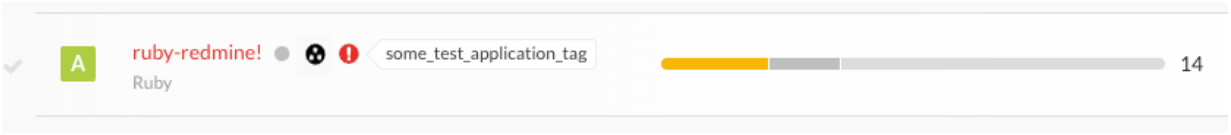
For merged applications, the Open Vulnerabilities column in the Applications list displays the number of vulnerabilities for all application modules in the primary application. The Applications list displays the primary application but not the modules in the primary application.

Example:

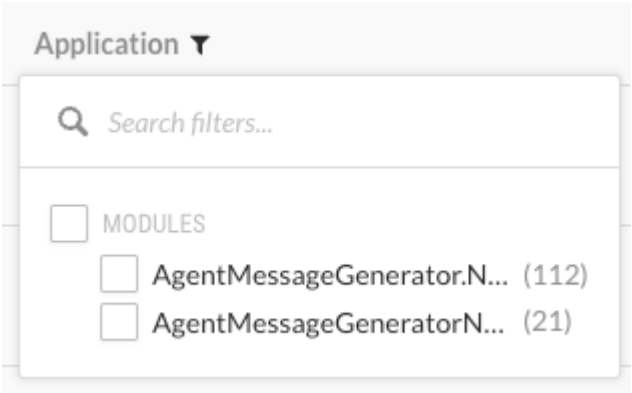
Before you merge applications, the Open Vulnerabilities column looks similar to this:



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 525)

View vulnerability rates

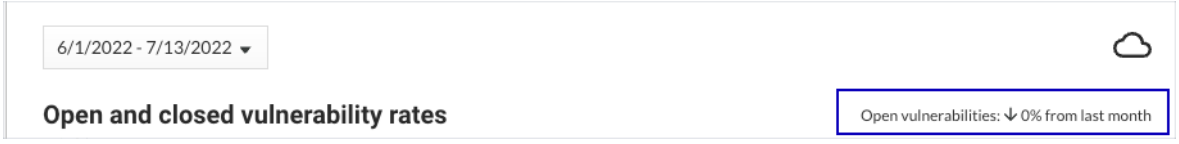
The chart for open and closed vulnerability rates shows trends for the vulnerabilities for all applications in your organization.

Steps

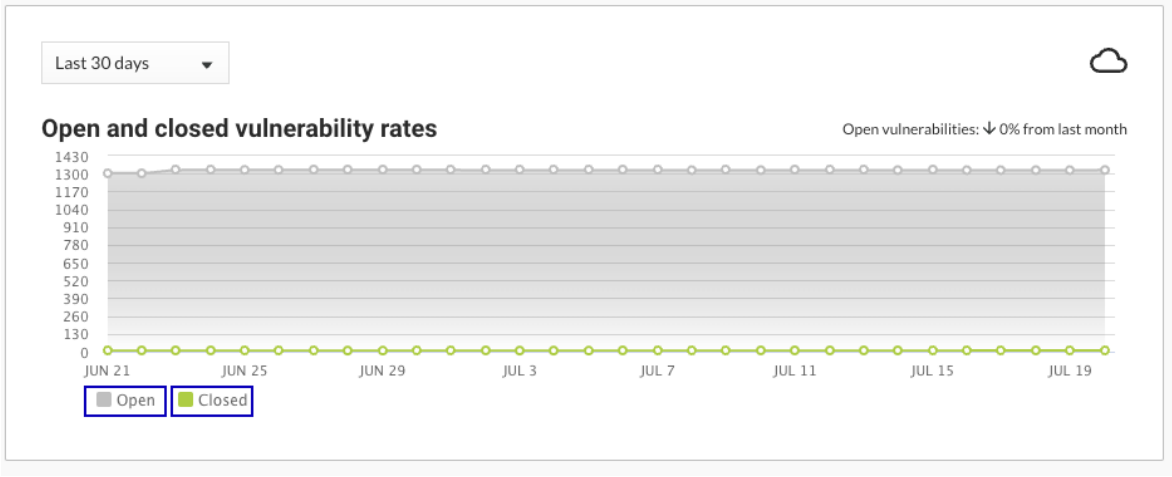
1. Display the new dashboard by selecting **View your new dashboard**.

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.
4. To view the percentage of change during the selected time frame, look at the value at the top of the chart

5. 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 525). 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:

1. Select **Vulnerabilities** in the header.
2. 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.

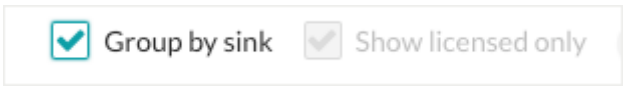
Grouping vulnerabilities reduces the number of entries that the vulnerability list shows. You can still see the data for individual vulnerabilities in a group.

Before you begin

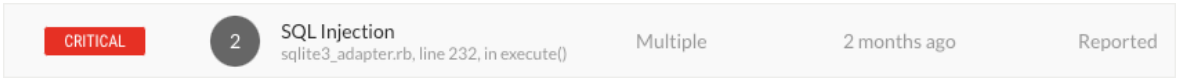
Group by sink applies only to vulnerabilities for applications with an Assess license. If you group by sink, **Show licensed only** is selected automatically.

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:



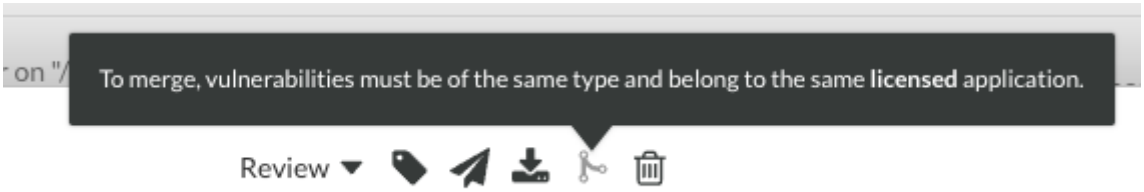
The number indicates the number of vulnerabilities in the group.
The values for the Severity, Application, and Status columns change to **Multiple**, If the vulnerabilities in a group have different severities, apply to different applications, or have different statuses.

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.

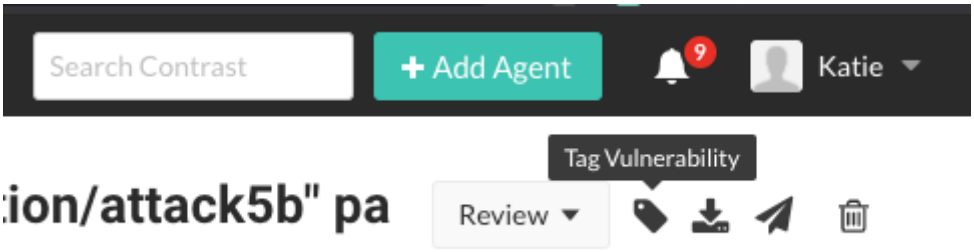


4. In the window that appears, select the vulnerability that you want to represent the merge.

Add a tag to a vulnerability

You can tag vulnerabilities to better organize vulnerabilities and improve search in Contrast. To do this:

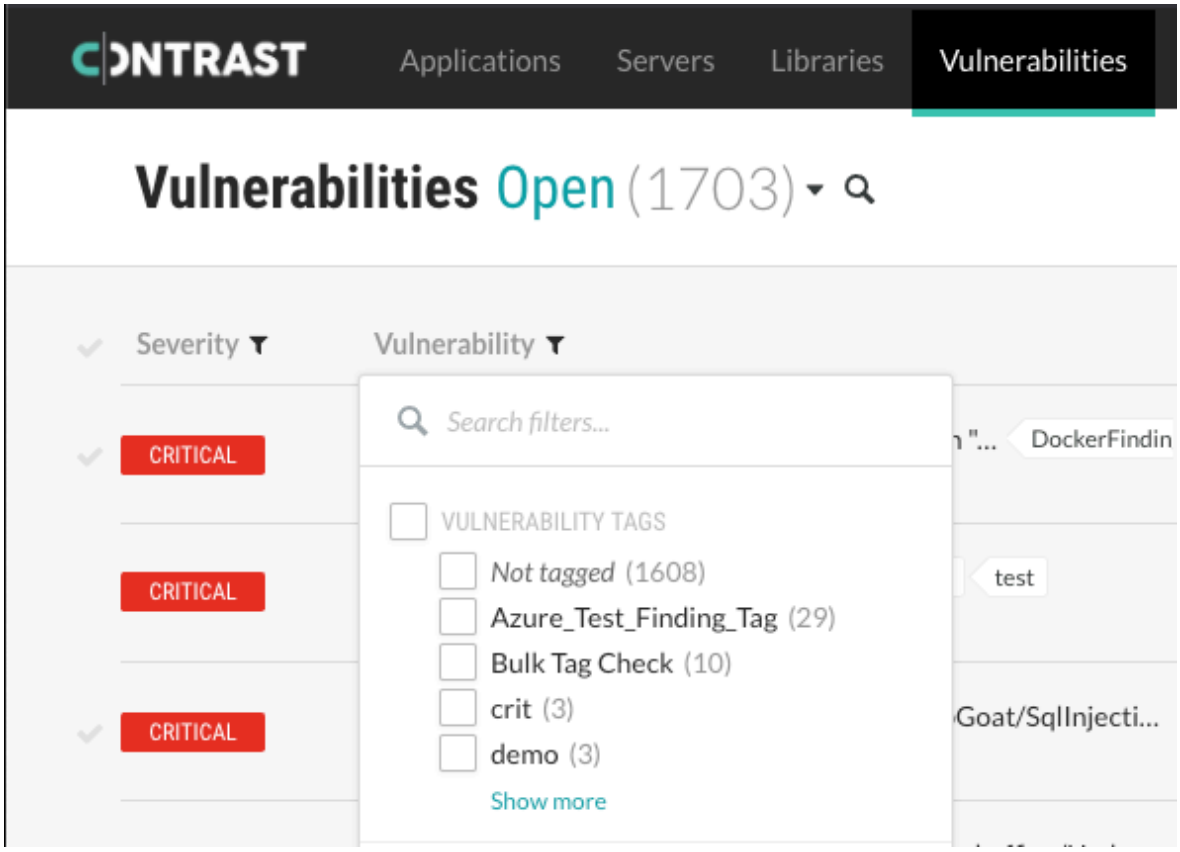
1. Select **Vulnerabilities** in the header, then hover over the row in the grid for the vulnerability you want to tag.
2. In the far right column, select the **Tag** icon. This option is also available from the vulnerability details page in the top right corner.



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:

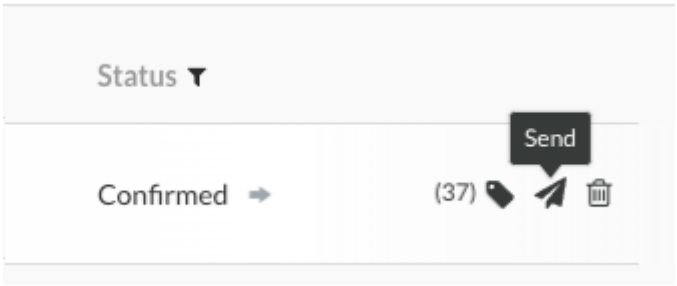
- Send vulnerability data to other members of your organization.
- Plan and maintain timely patching to prevent attacks.
- Streamline workflows by sending vulnerability information directly to your bugtracking tool.
- Receive notifications of any new high or critical vulnerabilities in your application.

Before you begin

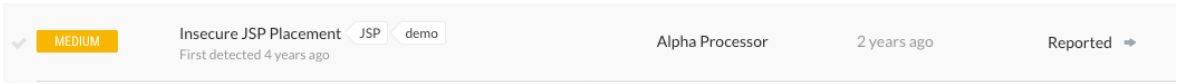
Be sure Contrast is integrated with at least one bugtracker (page 553) tool.

Steps

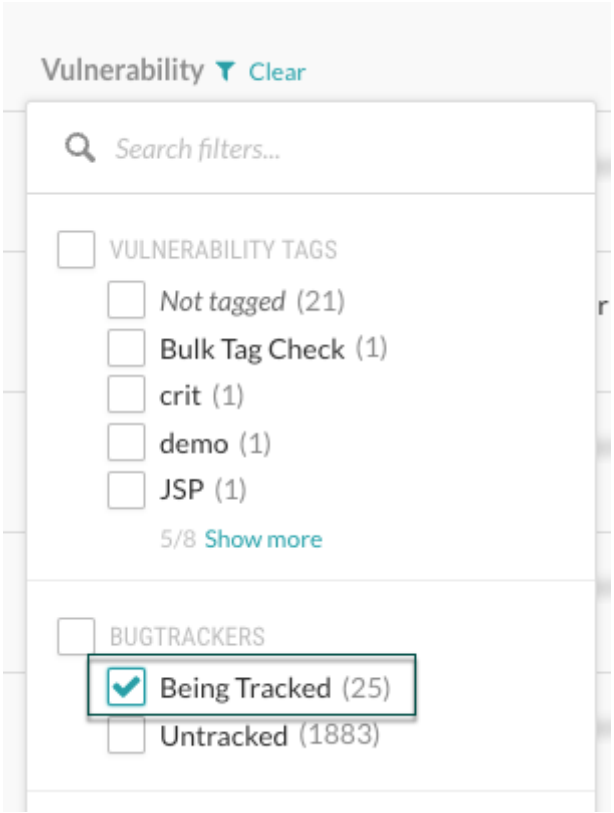
1. Select **Vulnerabilities** in the header.
2. To track a single vulnerability, hover over the end of the row for the vulnerability you want to track.
3. In the far right column, select the **Send** icon.
   This option is also available from the vulnerability details page.



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 534) to a CSV or XML file for custom processing, or use the API to gather data outside the web interface.

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 602) 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 602) or Application exclusions (page 620) 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 445) to learn more about a vulnerability using the build number.
     Add this property to the `-javaagent` command:

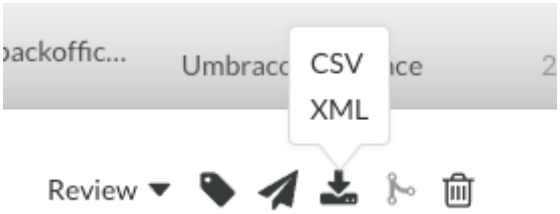
     ```
     -Dcontrast.override.appversion
     ```

     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 **Vulnerabilties** 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).



3. The export will download to your desktop. Exports contain the following data fields 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.

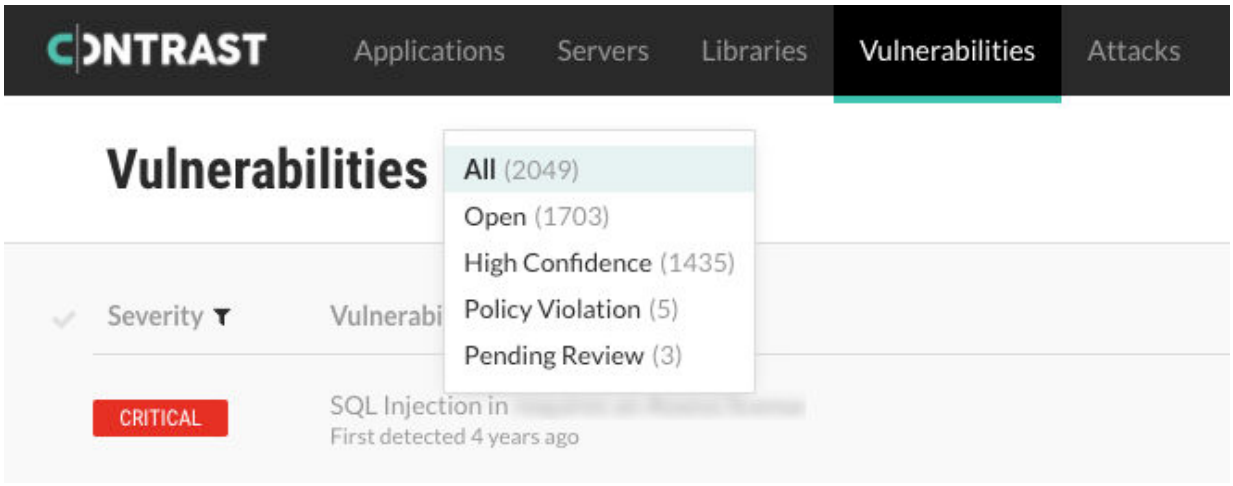
```
curl \
    -H "Authorization: $(echo -n $username:$servicekey |
 base64)" \
    -H "API-Key: $apikey" \
    https://app.contrastsecurity.com/Contrast/api/ng/$orgid/
orgtraces/filter?expand=request | \
    jq -r '.traces[] |
 {uuid: .uuid, protocol: .request.protocol} |
 [.uuid, .protocol] | @csv'
```

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
Reported	This is the default status of a vulnerability after it is discovered by Contrast. The vulnerability in this application could possibly be exploited.
Confirmed	Confirm that the vulnerability is a true finding by reviewing the source code or exploiting it.
Suspicious	The vulnerability appears to be a true finding based on the details provided, but requires more investigation to determine its validity.
Not a problem	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 **Reported** if found again. • **Attack is defended by an external security control:** There is another component in the environment, such as a WAF, which will prevent this vulnerability from being exploited. • **False positive:** This vulnerability was reported incorrectly. Contact Support to figure out why Contrast flagged this trace as a vulnerability. • **Goes through an internal security control:** There is custom, corrective code inside the application that will prevent this vulnerability from being exploited. • **URL is only accessible by trusted power users:** This vulnerability may only exist in specific environments, such as test, and may not exist in production environments. • **Other:** 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 replace **Other** with a custom value (page 537) that explains why the vulnerability is **Not a problem**.
Remediated	The vulnerability has been fixed by changing source code or config files within the application.
Fixed	The vulnerability has been fixed by changing the source code or because of a reason given under the **Not a problem** status. A vulnerability set to this status will not revert back to **Reported** if found again. (This option is only available to administrators.)
Remediated-Auto-verified	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 vulnerability policy (page 605), it will automatically be set to **Remediated-auto-verified**.

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.

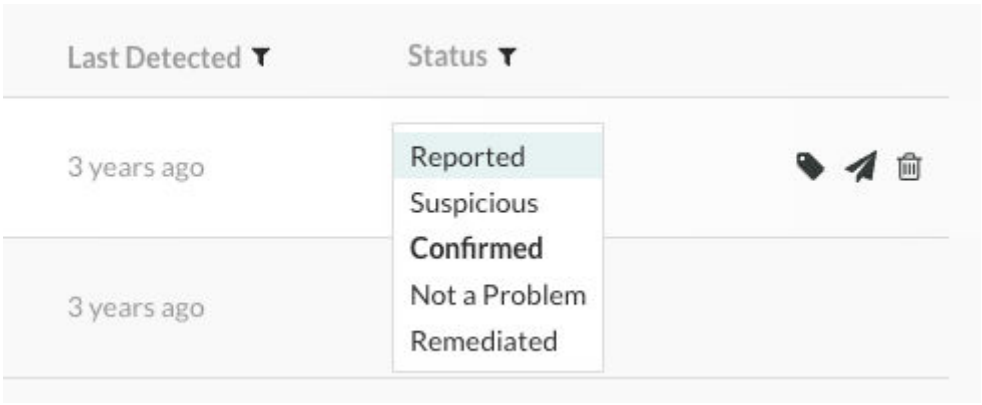


If the agent reports a vulnerability and Contrast has never seen it before, Contrast creates a new entry for the vulnerability. If that vulnerability already exists, Contrast updates the existing entry, issue count and number of days since it was last detected. All vulnerabilities will be reopened with the same pre-existing status, except those that were previously set to **Remediated** or **Remediated-Auto-verified**. Those will be reopened as **Reported**.

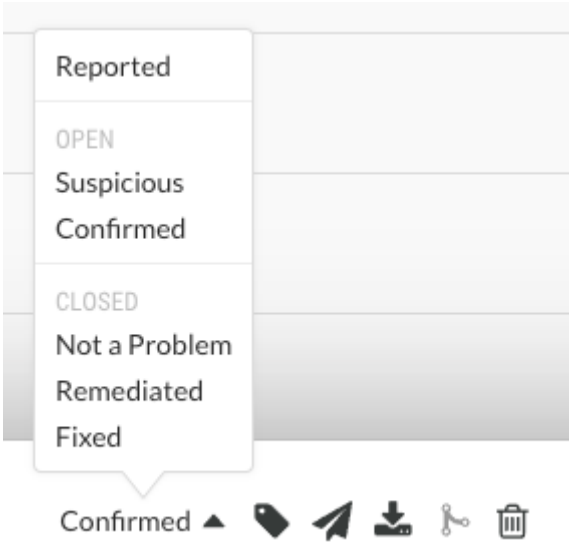
Edit vulnerability status

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.

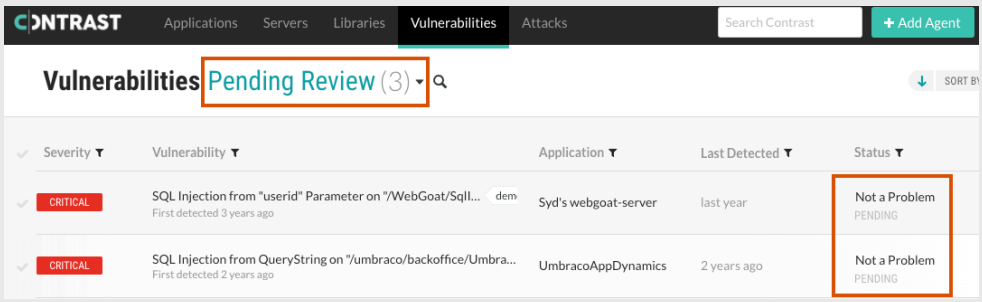


3. Select a new status according to these criteria (page 535).

NOTE

An Organization Administrator can require approval (page 662) before some vulnerabilities can be closed.

If this is the case, you will be required to submit a reason for your your status change, and it will be set to **Pending** until it can be reviewed by an 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 537) that vulnerabilities are **Not a problem**.

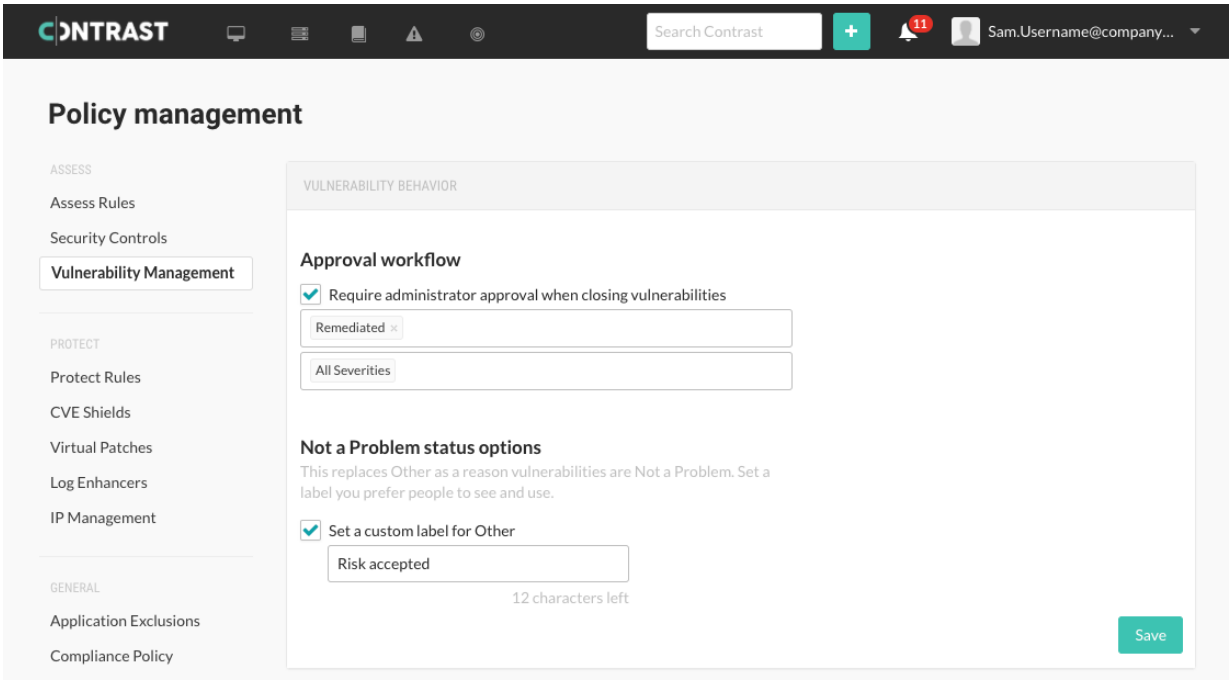
Set a custom reason that vulnerabilities are Not a problem

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 535) 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.



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.

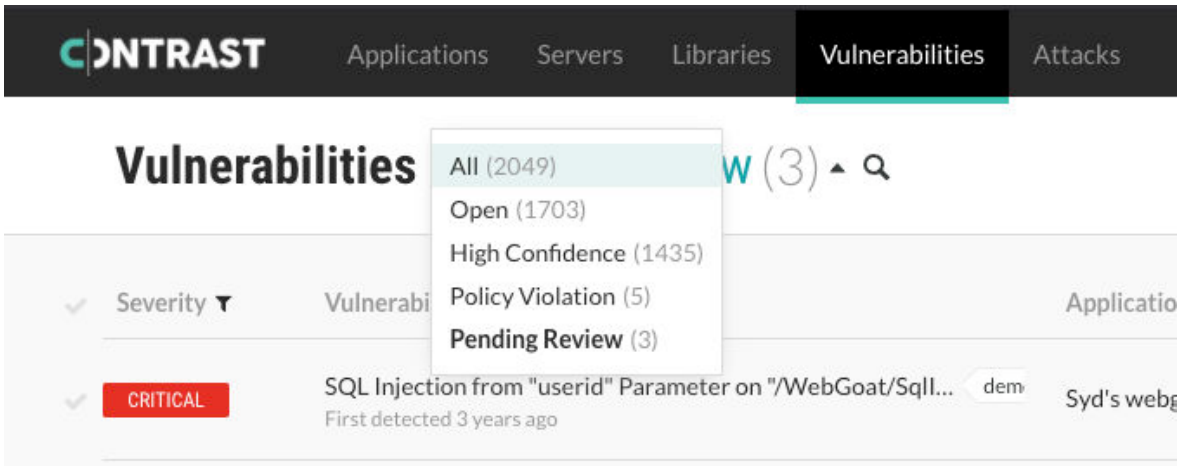
Review pending vulnerability status changes

If an Organization Administrator has required approval for a particular vulnerability (page 662), the status (page 535) 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.)

Attacks

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

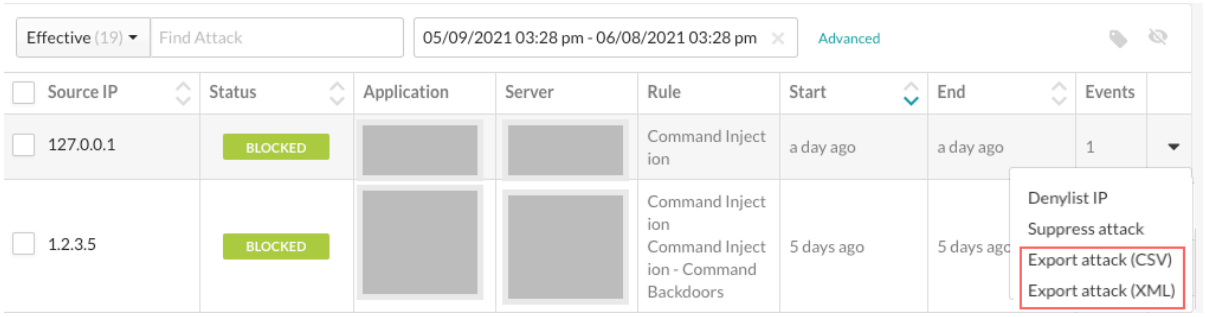
When Contrast detects multiple attack events from the same IP address within 30 minutes, Contrast groups these events together as an attack. If Contrast sees new events from the same IP address after you fix the code, Contrast shows a new attack.

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 thirty days before removing it. To keep attack data for a longer amount of time, do the following:

- Output to syslog (page 486)
- Set up a generic webhook (page 573)
  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.



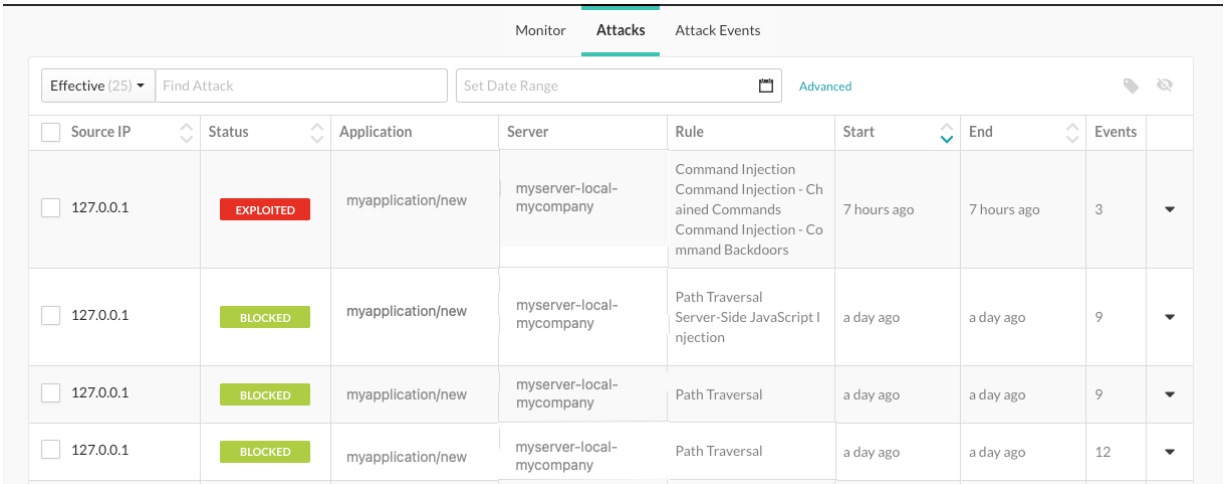
Tasks

In Contrast, you can:

- View attack details (page 540) such as which application and server was attacked and the location in the code where the attack occurred.
- Manage attacks (page 543) by taking actions on attacks and attack events. For example, you can configure a Protect rule for specific attack events.
- Monitor attacks (page 542) in an overview of current and past attacks..

View attacks

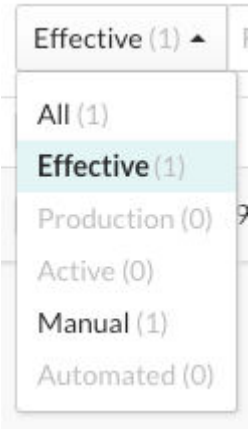
The Attacks list shows all attacks that have occurred in an organization.



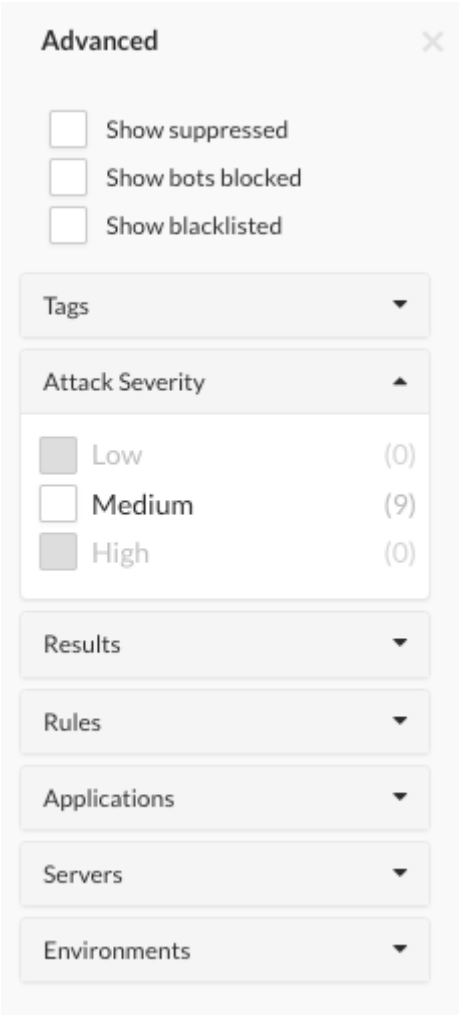
Steps

1. Select **Attacks** in the header.
2. To view all attacks that occurred in your organization, select the **Attacks** tab.
3. To filer 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.

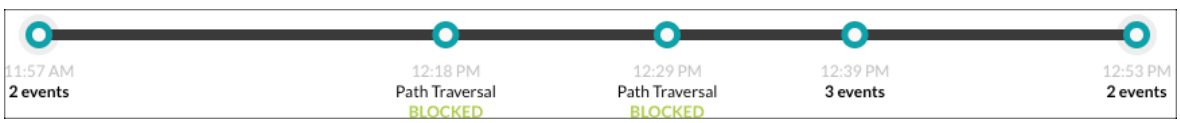
- **Automated:** Shows attacks that have more than 20 requests per second. It could indicate that a malicious bot is generating attacks.



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.



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

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 625), 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

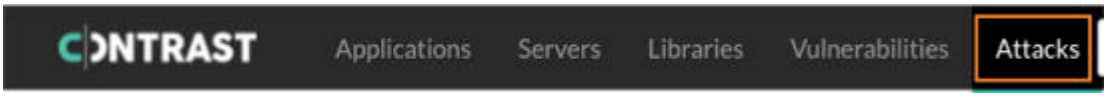
Before you begin

Ensure that Protect is enabled on the servers that host your applications. The Contrast header displays Attacks only when Protect is enabled.

Steps

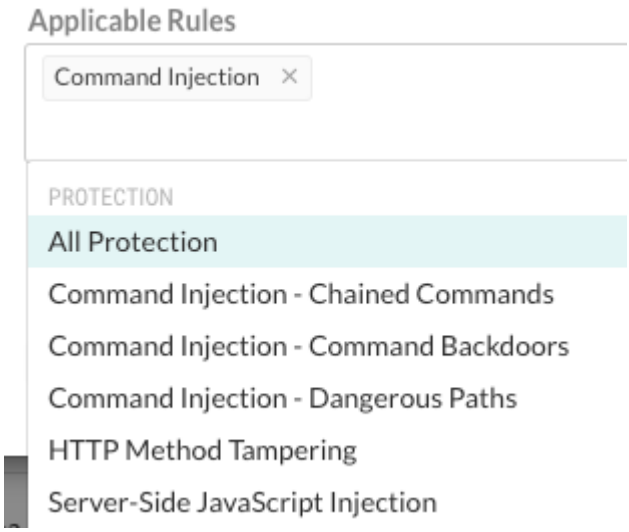
To take action on attacks and attack events, use the following procedures.

1. View attacks or attack events.
   a. Select **Attacks** in the header.

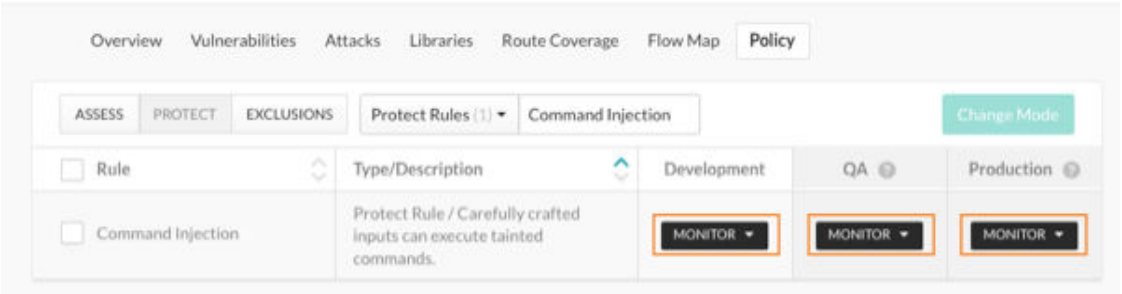
   b. Select the **Attacks** tab or the **Attack Events** tab.
2. (Optional) Tag attacks or attack events.
   Tagging attacks or attack events lets you organize them for better search results.
   a. Select one or more attacks or attack events.
   b. Select the tag icon ( ) above the list.
   c. In the Tag 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. Blocking an IP address prevents unwanted activity from a specific IP address in the future.
   To create a Protect rule (page 609) that blocks the IP address for a specific attack or attack event, use the following procedure:
   a. At the end of an attack or attack events row, select the arrow.
   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 620) lets you exclude certain applications, or parts of them, from security analysis.
   This option is available if you are using Java, .NET Framework , or .NET Core agents.

a. In the Attack Events grid, at the end of an attack events row, select the arrow.
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**.

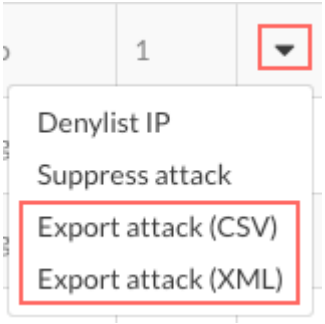
6. Create a virtual patch (attack events):
Virtual patches (page 617) are short-term, custom defense rules that defend against specific, newly-discovered vulnerabilities in your code.
a. In the Attack Events grid, at the end of an attack events row, select the arrow.
b. Select **Create Virtual Patch.**
c. In Add Virtual Patch (page 617), enter the details for the virtual patch.
d. Click **Save**.
7. Configure Protect rules (attack events):
Protect rules (page 609) let you monitor or block specific kinds of cyber-attacks in application environments.
a. In the Attack Events list, at the end of an attack events row, select the arrow.
b. Select a rule.
c. As needed, change the modes configured 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 486)

- Set up a generic webhook (page 573).
  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

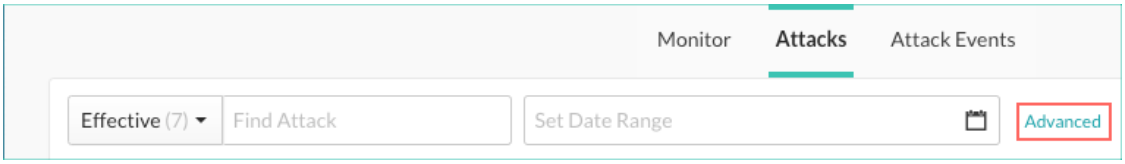
To make it easier to find specific attacks or attack events, use tags.

Before you begin

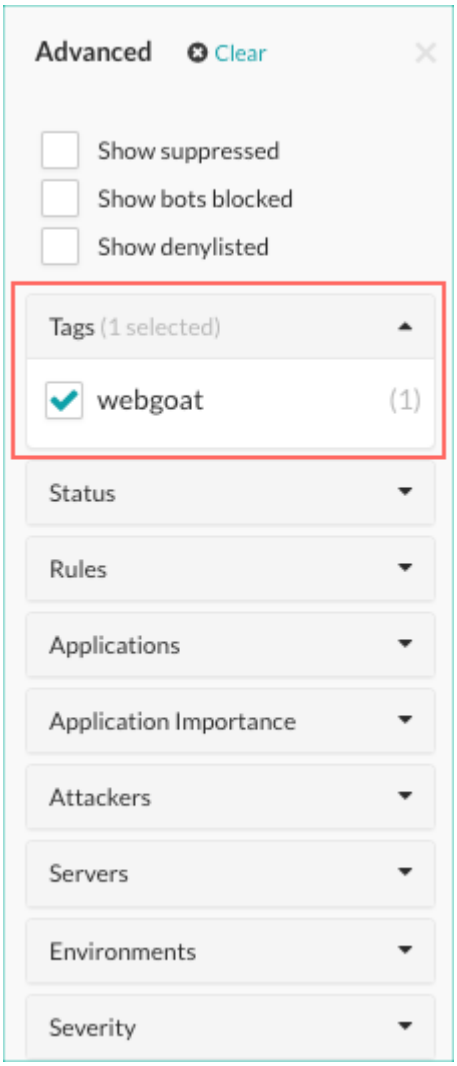
- An Organization Editor or Organization Administrator role is required.

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.



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.

Run attack scripts

To see how Contrast captures attack data, you can run an attack script using Nikto, an open-source web server scanner.

NOTE
To run an attack script you must have a Contrast agent installed (page 30) and an application with Contrast Protect enabled (page 629).

To run an attack script:

1. Run `./nikto.pl` in your terminal to make sure Nikto is configured correctly. If it is, you'll see a default help message return.
2. In Contrast, ensure that the IP address of the machine running Nikto isn't denylisted.
3. In the terminal, go to the *program* directory.
4. Initiate a scan by running

```
./nikto.pl -useragent “MyAgent (Demo/1.0)” -h http://www.your-site.com
```

NOTE
If your web application has its files under a certain directory, use the `-r` option to prepend a directory.

5. Once the script has finished running, Contrast will alert you about a new attack with an in-app notification and email.
6. Select the alert or go to **Attacks** to see a summary of the attack. To learn more about an individual attack, select **Source IP**.

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 547)
- Compliance reports (page 548)
- DISA STIG viewer checklists (page 549)
- Software bill of materials (page 549)
- Vulnerability trend report (page 550)
- Organization statistics (page 552)

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.

NOTE
This report expires seven days after you create it. Contrast then 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

To run an attestation report:

1. Select **Applications** in the header.
2. Select an application in the Applications grid.

3. Click the **Reports** icon located at the top-right of the application's page.
4. Select **Generate Attestation Report** from the list.
5. In the window that appears, define the **Vulnerabilities**, **Environments**, and additional **Security Standards** that you want to include in the report.
   The default is to show all vulnerabilities and environments, but you can filter them by clicking the fields. 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.
   The following table outlines the categories that you can use to create a custom report.

Field	Default	Filter options
Vulnerabilities	All	• Status (Reported, Suspicious, Confirmed, Not a Problem, Remediated, Fixed, Remediated - Auto-Verified) • Severity (Note, Low, Medium, High Critical) • Assess Rules
Vulnerability details	None	Include vulnerability details
Environments	All	• Development • Test • Production
Security Standards	None	• DISA ASD STIG • IPA-7.0 • OWASP 2013 Top 10 • OWASP 2017 Top 10 • OWASP 2021 Top 10 • OWASP Top 10 API Vulnerabilities 2019 • PCI DSS - 2.0 • PCI DSS - 3.0 • PCI DSS - 3.2.1

6. Select **Generate**.
   Once generated a download link appears in the **Notifications** panel.
   If an application has more than 16,000 vulnerabilities, Contrast does not generate the report. Contrast displays an error message indicating that the application exceeds the 16,000 vulnerability limit.
7. Click the report link to download the PDF.

Compliance reports

Generate timestamped PDF reports of security issues that Contrast has identified while monitoring your application. You can choose from the following report types:

- **DISA ASD STIG:** DISA's Application Security and Development STIG reports the security posture as it relates to policy requirements for security programs and best practices for Information Assurance (IA)-enabled applications.
- **OWASP 2013 Top 10, OWASP 2017 Top 10, and OWASP 2021 Top 10:** The Open Web Application Security Project reports the problems that are “worth fixing” or in the top ten list of flaws.
- **OWASP Top 10 API Vulnerabilities 2019:** The Open Web Application Security Project reports the unique vulnerabilities and security risks with APIs.
- **PCI DSS - 2.0, 3.0 and 3.2.1:** The Payment Card Industry Data Security Standard protects cardholder data in the event of a data breach. To achieve compliance, organizations must identify and remediate all critical vulnerabilities.

Each report includes a summary of the application's security status as well as details on each vulnerability and remediation guidance.

The report shows each vulnerability that's been discovered in your application, along with:

- Technical details
- Risk of an issue

- Remediation guidance
- Industry references
- The application's known vulnerable libraries
- A security scorecard

To run a compliance report:

1. Select **Applications** in the header.
2. Select an application in the Applications grid.
3. Click the **Reports** icon located at the top-right of the application's page.
4. Select **Generate Security Standards Report** from the list.
5. In the window that appears, choose the **Report Type**, **Vulnerability Status/ Severity** and **Vulnerability Tag** that you want to include in the report.
6. Click **Generate**.
   Once generated, the report will automatically download.

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

Before you begin

- A Contrast SCA license is required
- 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 the **Reports** icon ( ) located at the top of the application's list.
   c. In the dropdown, select **Generate Software Bill Of Materials (SBOM)** to generate and download a copy of the SBOM. Supports CycloneDx and SPDX standards.
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 `--sbom` command. Note that .NET support is currently limited for CLI. See CLI commands (page 513) for more information.

Vulnerability trend reports

Use the vulnerability trend reports to recognize the vulnerabilities your applications face and how well they're being managed.

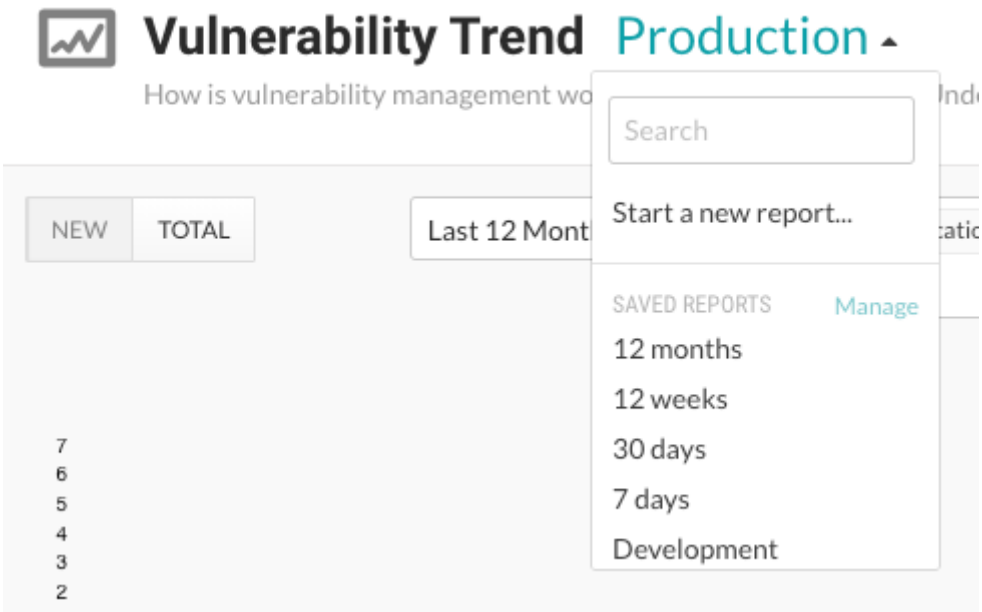
To view vulnerability trend reports:

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

Field	Default	Filter options
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 550), 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.

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 icon.

Cloud integrations

Run your application with Contrast while still deploying on your favorite PaaS.

AWS Elastic Beanstalk	Java agent with AWS Elastic Beanstalk
Google App Engine	Configure the Java agent for Google App EngineGoogle App Engine
Microsoft Azure	.NET and Azure App Service .NET and Azure Arm .NET and Terraform

	Red Hat OpenShift	Contrast Security containers
	VMware Tanzu (formerly Pivotal Cloud Foundry)	Configure Java for VMware Tanzu Configure Node.js for VMware Tanzu

Chat tools

Be the first to know about new vulnerabilities in or active attacks against your application as Contrast discovers these security issues in real-time.

	Microsoft Teams	Teams Integration (page 594)
	Slack	Slack Integration (page 595)

Code repository integrations

Add Github actions to your repository.

	Contrast Amazon Elastic Kubernetes Service Build Deploy	Contrast Amazon EKS Build Deploy
	Contrast Azure Kubernetes Service Build Deploy	Contrast AKS Build Deploy
	Contrast Azure Spring Cloud Deploy	Azure Spring Cloud Deploy
	Contrast SCA Action	Contrast SCA Action
	Contrast Scan Analyze	Contrast Scan Analyze

	Contrast Verify	Contrast Verify

Continuous integration and build tools

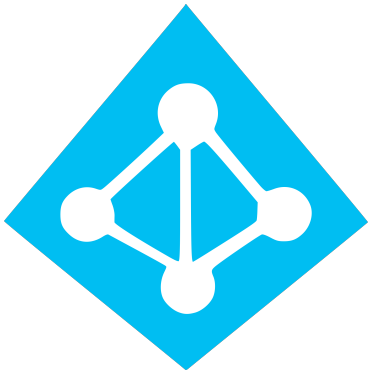
Add application security gates into your automated pipelines to prevent vulnerabilities from getting deployed into production environments.

	Azure DevOps	Azure Pipelines Extension (page 567) Azure Pipelines Marketplace
	Bamboo	Bamboo Plugin (page 571) Contrast Security for Bamboo Marketplace Contrast Bamboo Plugin Source Code
	CircleCI	Orb Registry Contrast Security Orb for CircleCI
	GitLab	How to integrate Contrast in a GitLab pipeline

	Gradle	Gradle Plugin (page 579) Gradle.org and Contrast Plugin Contrast Gradle Plugin Source Code Sample Project
	Jenkins	Jenkins Plugin (page 580) Contrast Continuous Application Security Jenkins Source Code
	Maven	Maven Plugin (page 54) Maven Central Repository Contrast Maven Plugin Source Code Sample Project

Enterprise and extensibility integrations

Use Contrast's SDKs and webhooks to build custom services and/or notify them upon the discovery of new vulnerabilities or attacks using webhooks.

	**Azure Active Directory**	Azure Active Directory single sign-on (SSO) integration

	Contrast CLI	Contrast CLI (page 511) NPM
	Java SDK	Contrast TeamServer Java SDK
	JavaScript SDK	Contrast SDK Javascript NPM
	.NET SDK	Contrast SDK .NET nuget

	Python SDK	Contrast SDK Python
	Webhooks	Generic Webhooks (page 573)

IDE plugins

Learn about your applications' vulnerabilities and receive remediation guidance while in your favorite development environment.

	Eclipse	Contrast Security for Eclipse Marketplace
	Intellij	Contrast Intellij Plugin Contrast Security for Intellij Marketplace

	Visual Studio	Visual Studio Plugin (page 596) Contrast for Visual Studio Marketplace
	Visual Studio Code	Visual Studio Code Plugin (page 597) Contrast Security Plugin Marketplace
	Visual Studio for Mac	Visual Studio for Mac Plugin (page 598) Extension file

Incident management systems

Be confident that on-call personnel will be equipped to take necessary action against attacks on your application.

	PagerDuty	PagerDuty Integration (page 594)

	Splunk on-call (formerly VictorOps)	VictorOps Integration (page 596)

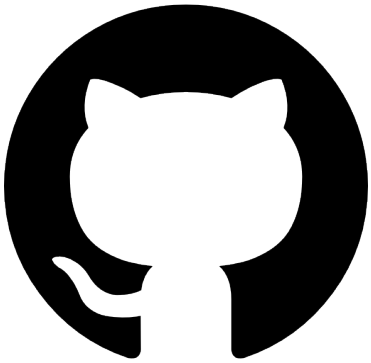
SIEM tools

Receive operational insights into application security threats from Contrast's instrumentation activities directly in your security information and event management (SIEM) tool.

	Azure Sentinel	Contrast Protect Azure Sentinel Solution Azure Sentinel
	Datadog	Source Code
	Splunk	How to integrate Contrast with Splunk

	Sumo Logic	Source Code

Work tracking platforms

	Azure Boards	Azure Boards Integrations (page 563)
	Bugzilla	Bugzilla Integration (page 572)
	**Github**	GitHub Integration (page 578)

	Jira	Jira Integration (page 589)
	Kenna Security	Kenna toolkit Running Contrast Security tasks
	Rally (formerly CA Agile Central)	Agile Central Integration (page 566)
	Solutions Business Manager (formerly Serena)	Serena Business Manager Integration (page 595)
	ThreadFix	Contrast Remote Provider (ThreadFix) Contrast Remote Provider

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.

You will need:

- 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.
- For more, see Microsoft's Azure Boards documentation.

See also

- Connect with Azure Boards (page 564)
- Automatically create tickets (page 564)
- Two-way integration (page 565)

- Set personal access tokens (page 565)

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 two-way integration (page 565) (to automatically update vulnerability status in Contrast) or automatic ticket creation (page 564) with Azure Boards.

See also

- Automatically create tickets (page 564)
- Two-way integration (page 565)
- Set personal access tokens (page 565)

Automatically create tickets

You can **automatically create tickets** every time Contrast discovers new vulnerabilities.

Steps

1. In the Azure Boards integration panel (page 563), 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 564)
- Two-way integration (page 565)

- Set personal access tokens (page 565)

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 563), 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 565)
- Connect to Azure Boards (page 564)
- Automatically create tickets (page 564)

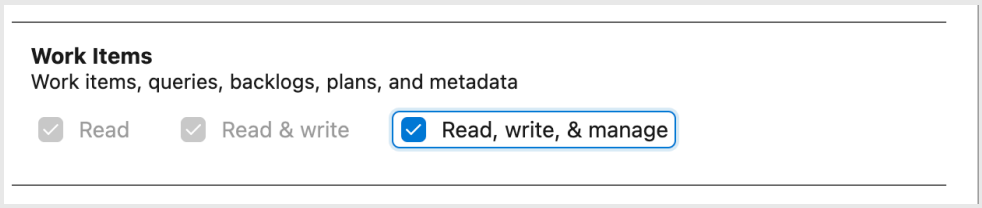
Set Azure Boards personal access tokens

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.



5. Click **Create** to finish and save your new personal access token.

See also

- Connect with Azure Boards (page 564)
- Automatically create tickets (page 564)
- Two-way integration (page 565)

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.
5. 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.

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

To install and configure the Azure Pipelines extension:

1. Follow the Microsoft instructions to install the extension **Contrast Integration**.

Follow the Microsoft instructions to install the extension **Contrast Integration**.
2. 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.
3. In the **Pipelines** section of the settings menu, select **Service connections**.
4. Select **New Service connection**and then **Contrast Server Connection**.
5. Complete all the fields with required data from your personal keys (page 436).

NOTE
Your Contrast URL should not include */Contrast* at the end; only the host is required.

Configure a task in the Azure Pipelines extension

To configure a task in your Azure Pipelines extension for a release or a build pipeline:

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. Click on the + button next to your agentless job, and add the **Contrast Assess - Application Vulnerability Detection** 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. Select one of your applications from the **Application** menu.
7. 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.
8. 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. To add a release gate with the Azure Pipelines extension:

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 **Contrast Assess - Application Vulnerability Detection**.
5. Select **New** next to the service connection dropdown to create a Contrast service connection. Fill in all the fields and select **OK**.
   Select **Refresh list**, then select your newly created connection.
6. Click over 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. 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.
   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.

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).

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 133). 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>
```

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\ContrastProfiler-32.dll" />
            <EnvironmentVariable Name="COR_PROFILER_PATH_64" \
Value=".\contrastsecurity\ContrastProfiler-64.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.

Integrate with Bugzilla

To integrate with Bugzilla:

1. Go to **Organization settings > Integrations** in the **user menu**.
2. Select **Connect** in the Bugzilla row.
3. In the window that appears, complete the Bugzilla properties fields:

Field	Description
Name	A name for the bugtracker entry. It will be displayed when sending findings to bugtrackers
Username	The username for the account connected with Bugzilla
Password	The password for the username specified
Host	The URL of the Bugzilla instance
Application	The application you would like to map to a Bugzilla product/component
Product	The product in Bugzilla to map to the application
Component	The component in Bugzilla to map to the application
Version	The version in Bugzilla to map to the application
Priority	The priority to use when exporting findings to Bugzilla

4. Select **Test Connection** to verify communication. This ensures that Contrast can communicate and authenticate with the Bugzilla instance, and verifies the existence of the specified **Product**, **Component** and **Version**.

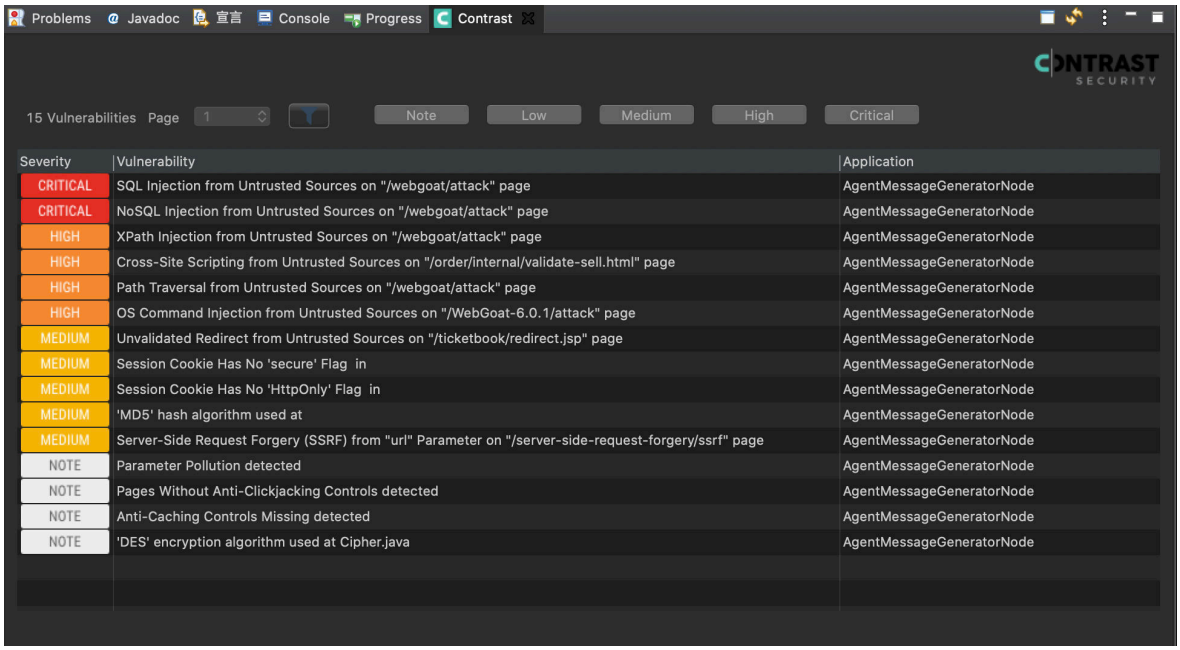
Eclipse

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.

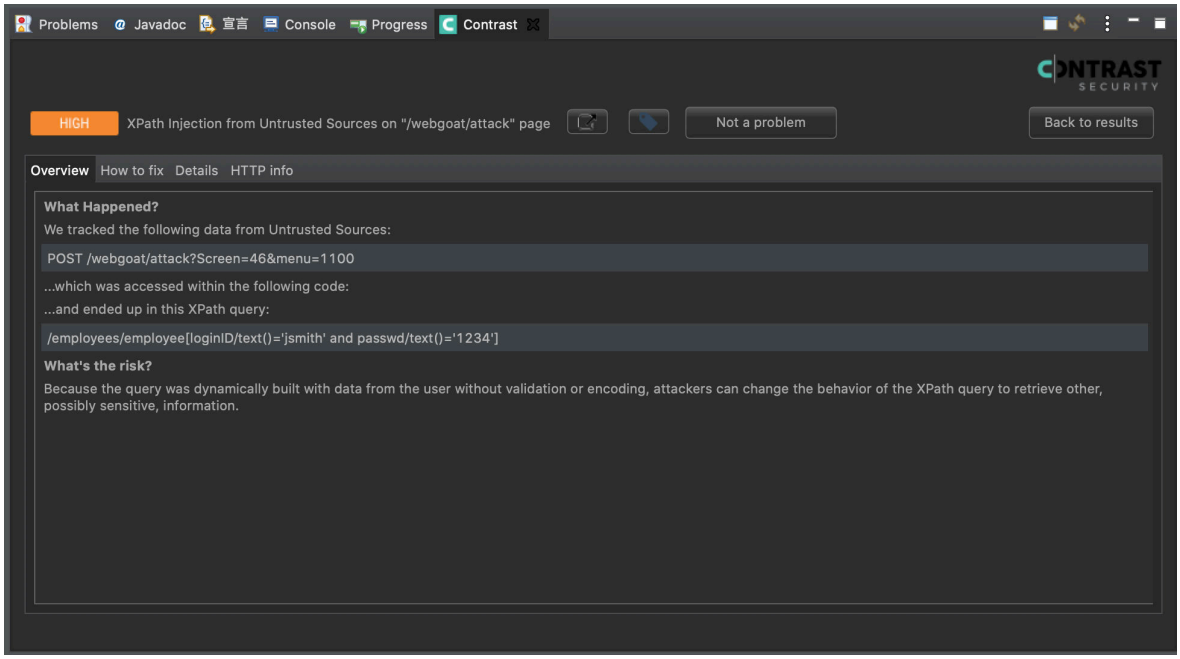
This plugin is available for Mac/OS, Linux, and Windows and supports the latest versions of Eclipse.

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 436) 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 on any URL that receives POST messages. You can add custom variables (page 575) to your payload like `$ApplicationName` and `$ServerId` when a Contrast event (page 578) triggers them.

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 575). For example:

```
{
   "title": "$Title",
   "message": "$Message"
}
```

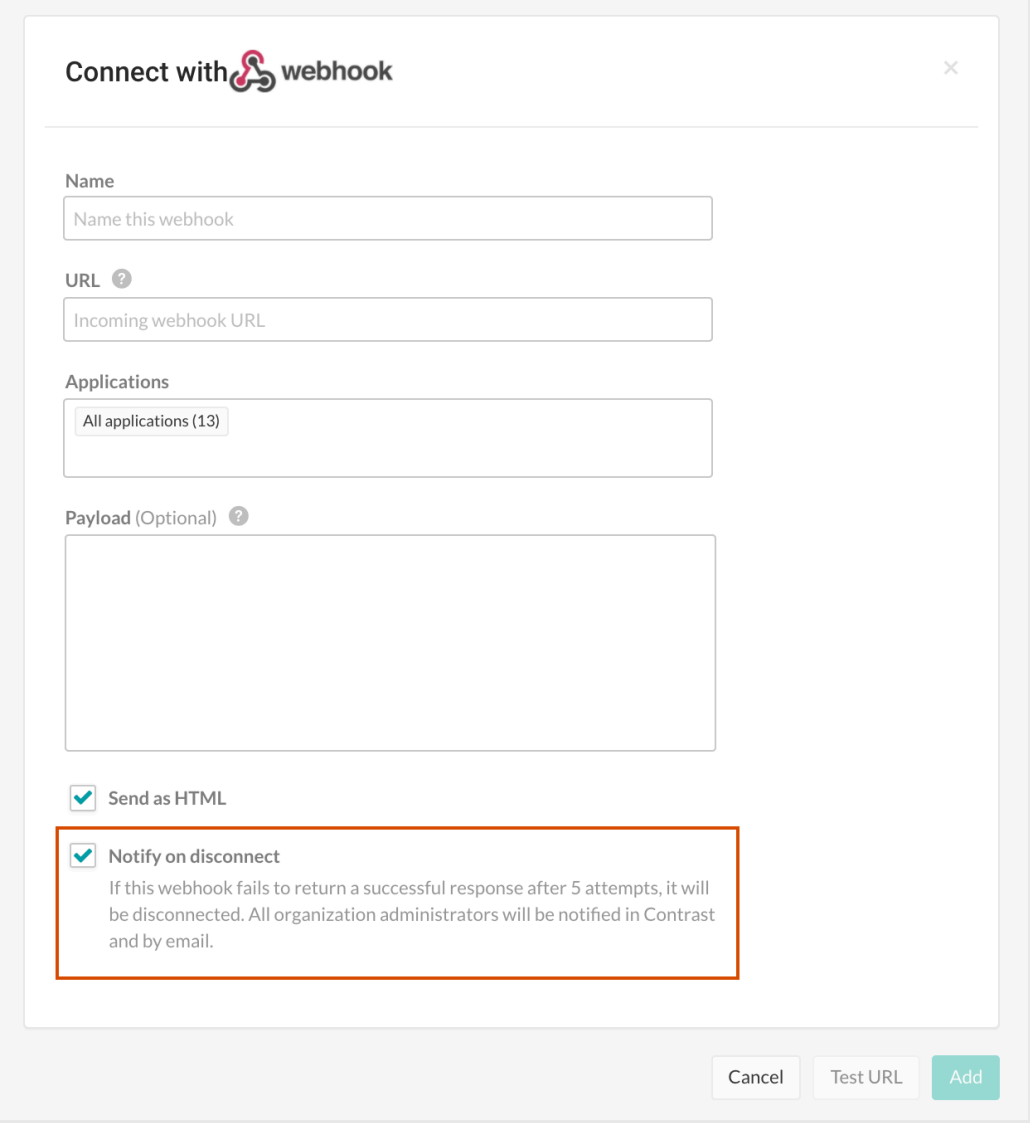
7. Select **Add**.

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**.



Generic webhook variables

You can customize your generic webhook (page 573) 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
General variables	
`$EventType`	The event type responsible for triggering the webhook For example: SERVER_OFFLINE
`$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
`$Title`	Always returns “Contrast Security”
Application variables	
`$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.
`$ApplicationTotalModules`	The number of modules your application has
Server variables	
`$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
Vulnerability variables	
`$Severity`	If this event is triggered by a vulnerability, this is the severity of the vulnerability
`$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 573) response with data from Contrast events such as `NEW_VULNERABILITY` and `SERVER_OFFLINE`. Each event contains general (page 575), application (page 575), server (page 575) or vulnerability (page 575) variables you can call in your payload request.

Event	Variables
`ATTACK_END`	General (page 575), Application (page 575), Server (page 575)
`ATTACK_EVENT_COMMENT`	General (page 575), Application (page 575), Server (page 575)
`ATTACK_UPDATE`	General (page 575), Application (page 575), Server (page 575)
`EXPIRING_LICENSE`	General (page 575), Application (page 575)
`NEW_ASSET` (if new application)	General (page 575), Application (page 575) and Server (page 575) (if new application)
`NEW_ATTACK_APPLICATION`	General (page 575), Application (page 575), Server (page 575)
`NEW_ATTACK_UPDATE`	General (page 575), Application (page 575), Server (page 575)
`NEW_ATTACK`	General (page 575), Application (page 575), Server (page 575)
`NEW_VULNERABILITY_COMMENT`	General (page 575), Application (page 575), Server (page 575), Vulnerability (page 575)
`NEW_VULNERABILITY`	General (page 575), Application (page 575), Server (page 575), Vulnerability (page 575)
`NEW_VULNERABLE_LIBRARY`	General (page 575), Application (page 575)
`SERVER_OFFLINE`	General (page 575), Server (page 575)
`VULNERABILITY_CHANGESTATUS_CLOSED`	General (page 575), Application (page 575), Server (page 575), Vulnerability (page 575)
`VULNERABILITY_CHANGESTATUS_OPEN`	General (page 575), Application (page 575), Server (page 575), Vulnerability (page 575)
`VULNERABILITY_DUPLICATE`	General (page 575), Application (page 575), Server (page 575), Vulnerability (page 575)

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**.
5. 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.

6. Once a connection is made, select the **Applications** that you want to be available to this bugtracker.
7. 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.

8. 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.

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 436) 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
`contrastInstall`	Installs a Contrast Java agent to your local project. The plugin edits the `org.gradle.jvmargs` property in the *gradle.properties* file to launch the JVM with the Contrast agent. An application version, by which the vulnerabilities are filtered in the `contrastVerify` task, is generated during this task. The plugin generates the application version in the following order: • If your build is running in TravisCI, Contrast uses `appName-$TRAVIS_BUILD_NUMBER`. • If your build is running in CircleCI, Contrast uses `appName-$CIRCLE_BUILD_NUM`. • If your build is running in neither TravisCI nor CircleCI, Contrast generates one in the format `appName-yyyyMMddHHmm`. (Where `yyyyMMddHHmm` is the time of the build generation.)
`contrastVerify`	Checks for new vulnerabilities in your web application.

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 581) 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 582) or as a post-build action step (page 583).
   - Define vulnerability security controls for pipeline steps (page 583).
   - Optionally, a Contrast Organization Administrator define a job outcome policy (page 585).
3. Run a build (page 589).

See also

- Connect with Jenkins (page 581)
- Define security controls (page 582)
- Define a job outcome policy (page 585)
- Run a build (page 589)

Connect

Before you begin

- All Contrast Security integrations require an organization admin role (page 744) 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 436) 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 585) 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 582)
- Define a job outcome policy (page 585)
- Run a build (page 589)

Jenkins security controls

You can define security controls for Jenkins:

- At a system level (page 582),
- as a post-build action step (page 583),
- or for pipelines. (page 583)

Define security controls at a system level

After you define a connection (page 581) 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 583), or if you use a job outcome policy (page 585) 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 539) 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 aren't required, but can be helpful. For example, select **Confirmed** and **Suspicious** to only return vulnerabilities with an open status. Leave this blank if you don't want to filter vulnerabilities by statuses.
   You can add multiple vulnerability security controls, but the plugin will fail the job on the first bad condition. The plugin will 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 582) 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 582), 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
`profile`	Required	Contrast connection profile used to communicate with Contrast	`MyConnection`
`outputDirectory`	Required	This defines where to put the downloaded agent.	`env.WORKSPACE`

Parameter	Required	Description	Examples
`agentType`	Required if `applicationId` 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
`profile`	Required	Use `profile` to specify the connection used to communicate with Contrast.	Contrast Connection
`queryBy`	Required	Use `queryBy` to filter build-related vulnerabilities. For options 1, 2 and 4, this value must match the `contrast.override.appversion` 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): 1. `appVersionTag, format: applicationId-${BUILD_NUMBER}` 2. `appVersionTag, format: applicationId-${JOB_NAME}-${BUILD_NUMBER}` 3. `startDate` (This is the build timestamp. It only looks at vulnerabilities discovered after the build starts.) 4. `APPVERSIONTAG` (This is the job parameter or environment variable. Select this option if you want to specify your own text, then export APPVERSIONTAG as an environment property within your Jenkins job. Both `JOB_NAME` and `BUILD_NUMBER` are already available as Jenkins environment properties.)	1
`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 **All**	`xss`
`severity`	Optional	Defaults to **All**. Other options are **Critical**, **High**, **Medium** and **Low**.	**High**
  ```

Parameter	Required	Description	Examples
`appVersionTag`	Optional	The value that was passed to the `contrast.override.appversion` 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 585), 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'
```

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.

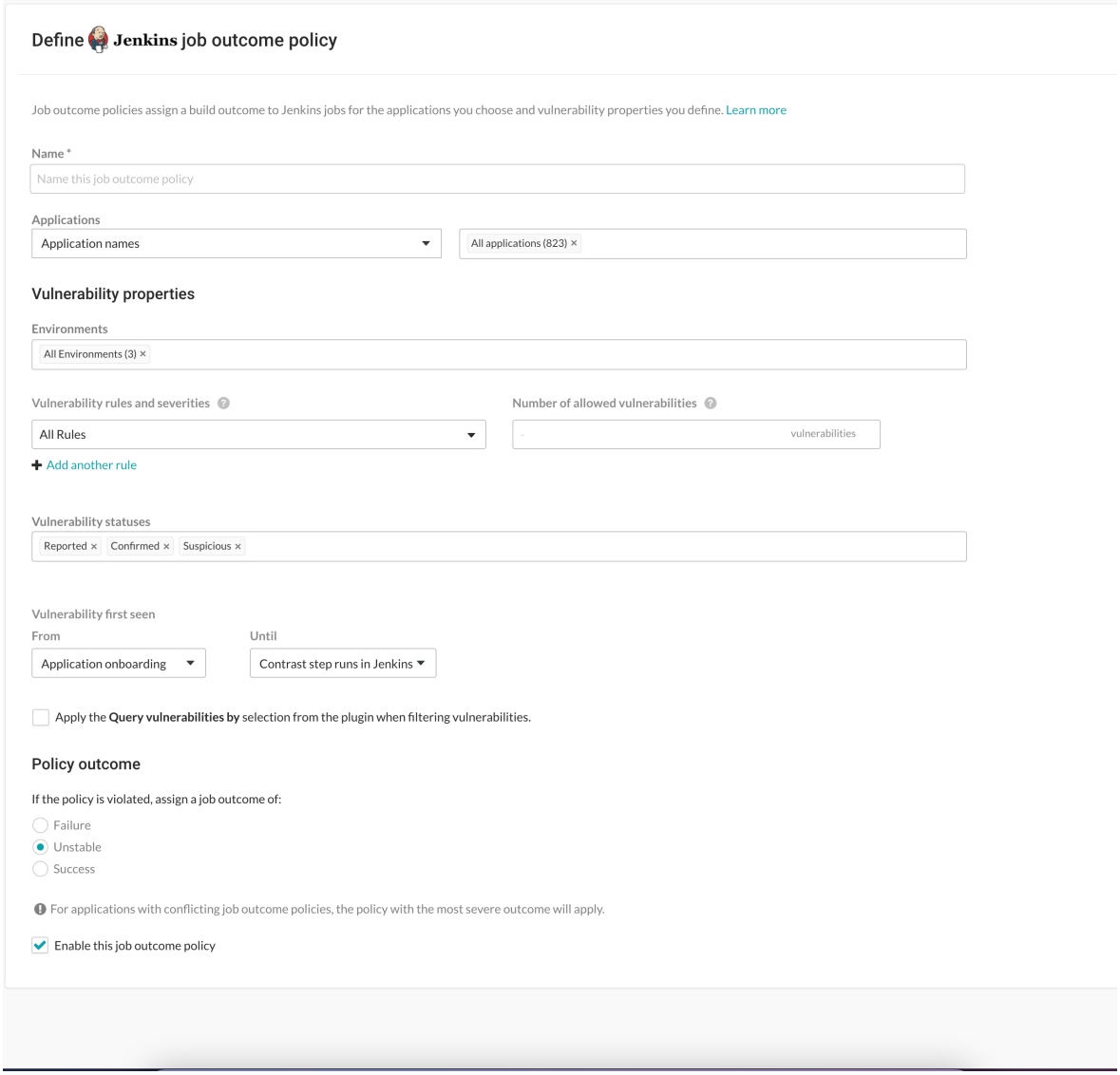
Before you begin

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 632), select **Integrations** in the left navigation.
2. In the **Jenkins** row, select **Add 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 442).
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 535) are determined by Contrast.

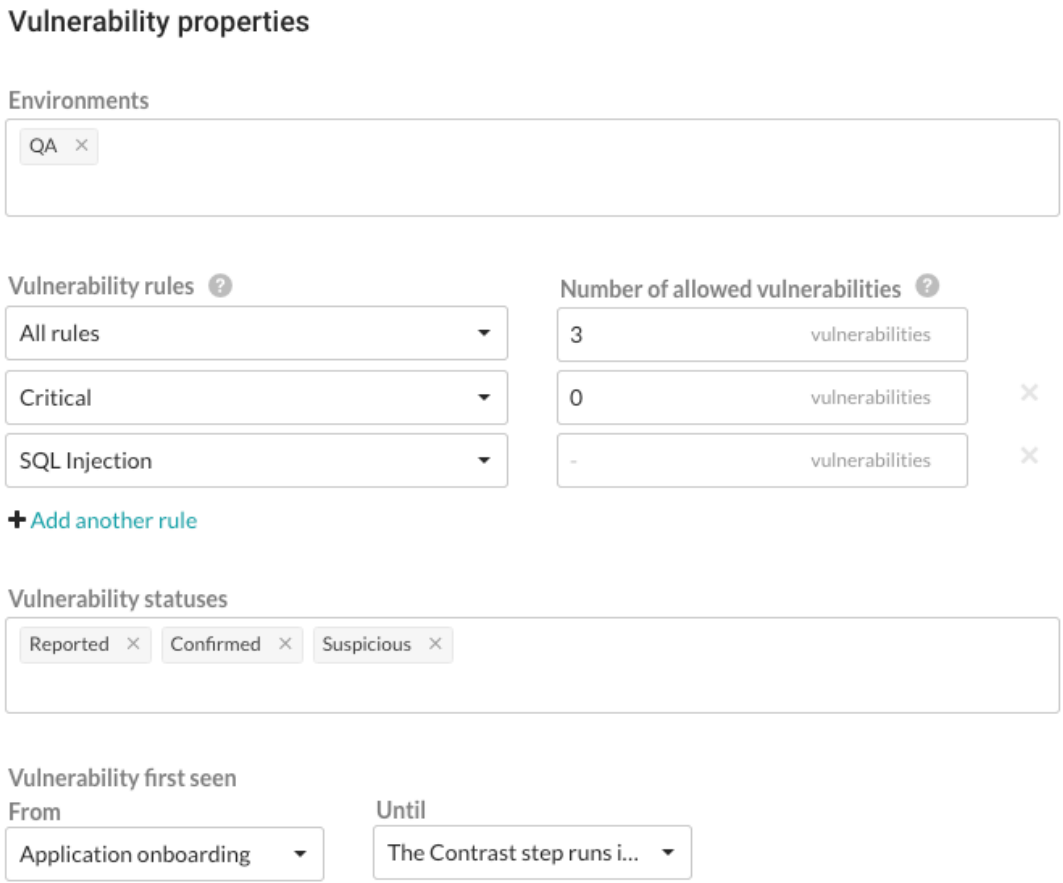
> **TIP**
> In most cases, you should only select open statuses like **Reported**, **Confirmed**, and **Suspicious** (rather than closed statuses like **Not a problem**, **Remediated** or **Fixed**). That way, Jenkins jobs won't fail or turn unstable due to vulnerabilities that developers have already 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 vulnerabilites 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 583) or pipeline step (page 584). 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.



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 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 589)
- Configure Jira for Assess (page 590)
- Configure Jira for Serverless (page 591)
- Manager Jira credentials (page 592)

Connect to Jira

To integrate Jira with Contrast:

Setup 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 590) or Jira for Serverless (page 591).

See also

- Configure Jira for Assess (page 590)
- Configure Jira for Serverless (page 591)
- Manage Jira credentials (page 592)

Configure Jira for Assess

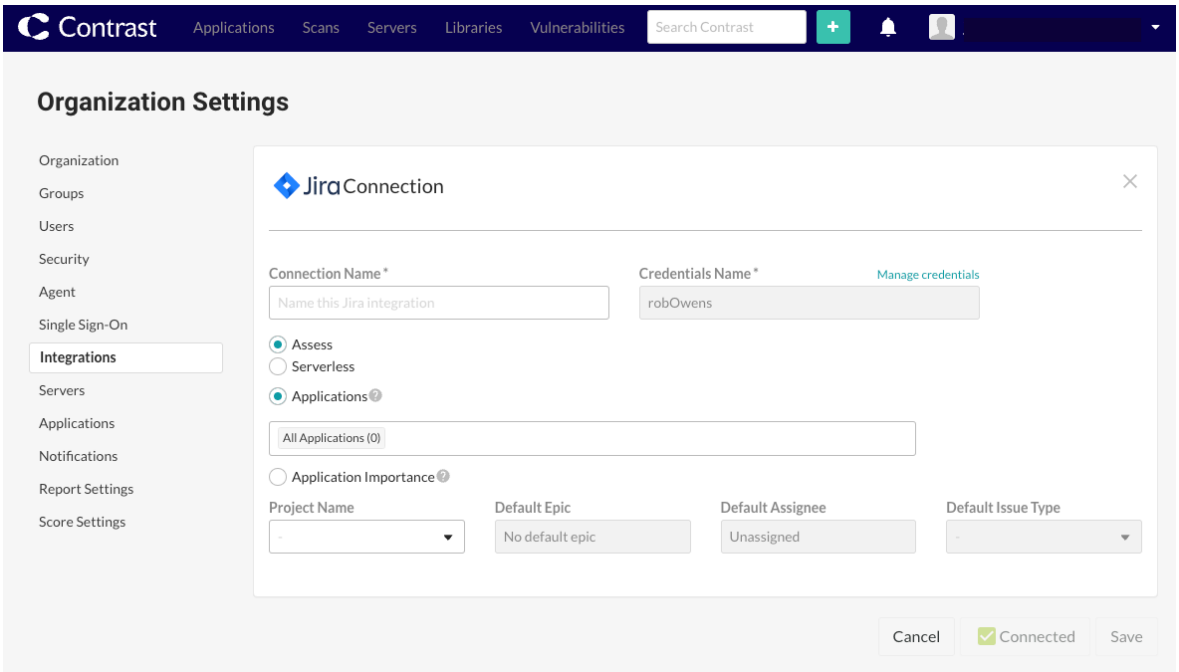
After testing your Jira connection, you can configure Jira to create tickets based on triggers you've set.

Before you begin

- Successfully connected with Jira (page 589)

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.



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 groom 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 option to **Enable two-way integration**, if you want to change vulnerability status in Contrast every time an issue closes or reopens in Jira. 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 registerer webhooks with non-https URLs. Therefore, Contrast on-premise users need to configure HTTPS (page 686) before attempting to enable Jira two-way integration.

4. 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.

5. Select **Save** and begin using your Jira integration. To remove the integration select **Delete configuration**.

Configure Jira for Serverless

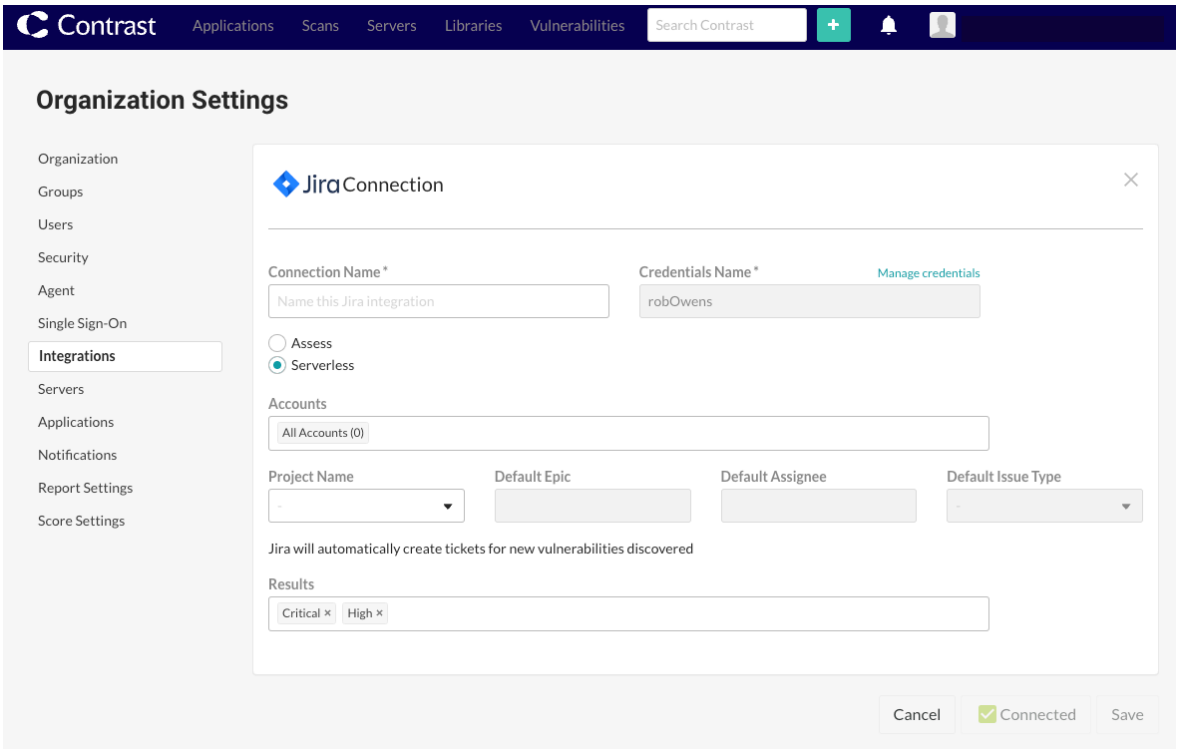
After testing your Jira connection, you can configure Jira to choose a subset of severities and result categories from which the integration should create a Jira ticket.

Before you begin

- Successfully connected with Jira (page 589)

Steps

1. After Contrast connects to Jira, select **Accounts** to configure the AWS accounts that from which results will have Jira tickets created.



2. Choose a Jira **Project** in which Contrast should create tickets.
3. Choose the type of **Results** for which Contrast should create tickets.
   - Permissions
   - Exploits
   - Dependencies (CVEs)
4. Set the **Default Issue Type** and **Default Assignee**.

> **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.

5. 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.
6. Select **Save** and begin using your Jira integration. To remove the integration select **Delete configuration**.

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 that 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, if you want. 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, use the dropdown to select a set of saved credentials.
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.

See also

- Jira integration (page 589)
- Connect to Jira (page 589)
- Configure Jira for Assess (page 590)
- Configure Jira for Serverless (page 590)

Contrast Maven plugin

Maven is a build tool that builds, packages, and tests your Java applications.

The Contrast Maven plugin can integrate Contrast Asses 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.
- **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 455)
- Install the Java agent (page 54)

Integrate with Microsoft Teams

Use the Microsoft Teams integration to receive notifications from Contrast in your configured Microsoft Teams instance.

To connect Microsoft Teams:

1. Go to your team in your Microsoft Teams account and choose the channel you want to send messages to.
2. Select the **More** icon.
3. Select **Connectors** from the menu.
4. In the **Incoming webhook** row, select **Configure**.
5. Enter a name for the Incoming webhook and click **Create**.
6. Copy the webhook URL. You will use this to set up the integration in Contrast.
7. Select **Save**.
8. In Contrast, under the **user menu**, go to **Organization settings > Integrations**.
9. In the Microsoft Teams row, select **Connect**.
10. Enter a name for the integration.
11. Paste the webhook URL copied from Microsoft Teams.
12. Select an application to enable notifications.
13. Select **Save**.

IMPORTANT

Contrast will disconnect this integration if it fails to return a successful response after 5 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 643).

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 Slack

With the Slack integration, you can receive notifications 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 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) and 2019.

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 436).

> **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.

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 436).
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 436).
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.

Administration

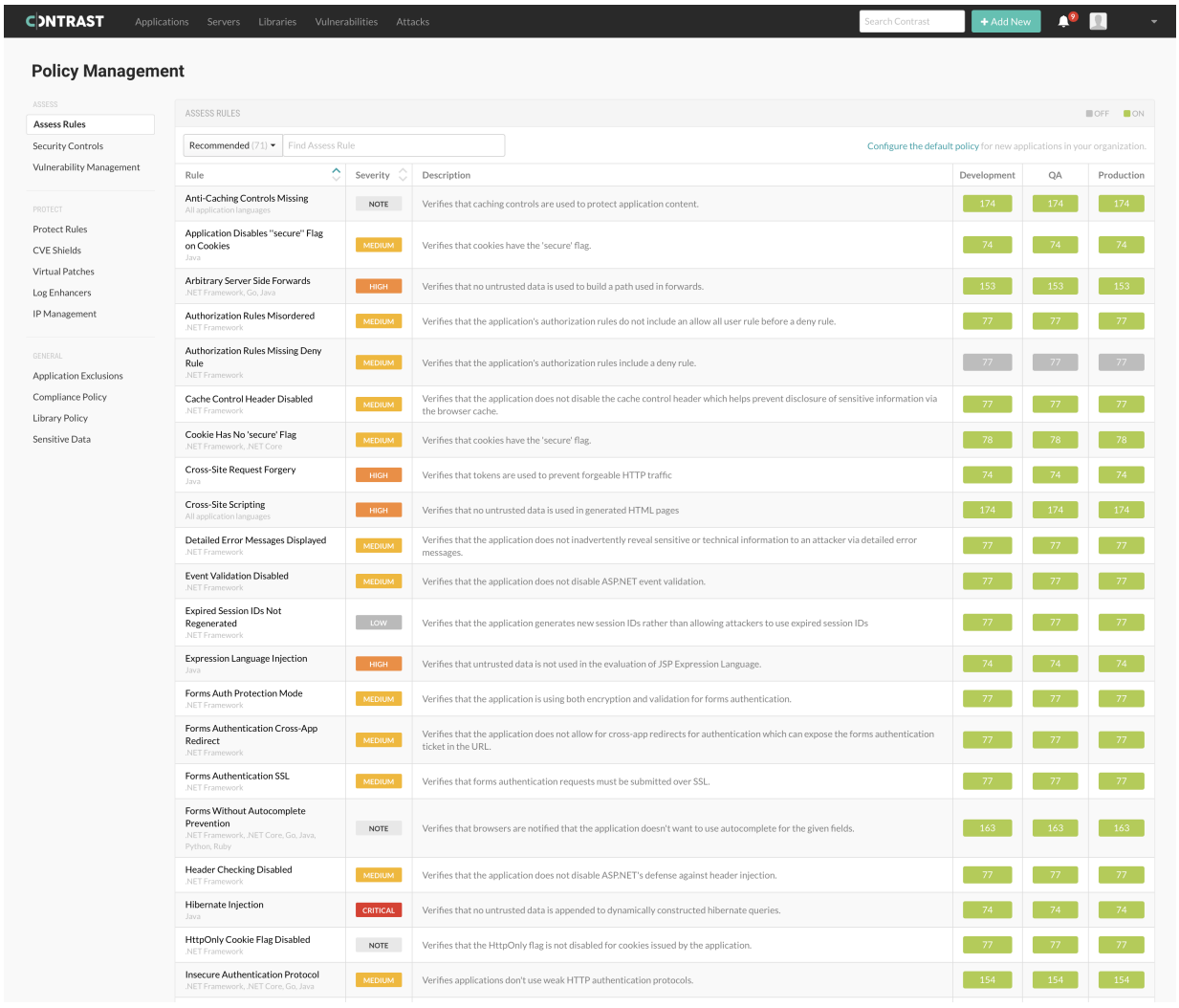
Users with different roles and permissions (page 742) have different levels of access in Contrast. This allows different individuals and teams in your business to best use Contrast for their responsibilities:

- Rules and policy administration: (page 600) RulesAdmins can create and edit policies. Editors can add applications and manage some content details like scores and notifications.
- Organization administration: (page 628) Organization Administrators can configure settings for a particular organization.
- System administration: (page 668) 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**.



Here you can manage:

- Assess rules (page 601)
- Security controls (page 602)
- Vulnerability policy (page 605)
- Set Protect rules (page 609)
- CVE shields (page 613)
- Virtual patches (page 617)
- Log enhancers (page 618)
- Block or allow IP addresses (page 624)
- Edit IP source names (page 625)
- Application exclusions (page 620)
- Compliance policy (page 623)
- Library policy (page 626)
- Sensitive data (page 628)

A user with RulesAdmin permissions can also:

- Instrument an application (page 30)
- Enable Protect (page 629)
- Approve or deny pending vulnerability status changes (page 538)
- Enable notifications (page 628)
- Set default scoring (page 646)

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 602).

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.

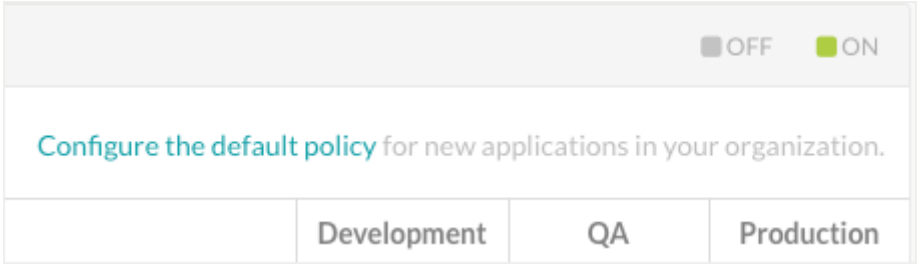
Use this procedure to change the default settings for Assess rules at an organization level. These settings apply to any new application that you add to a Contrast organization. These changes have no affect on existing applications in the organization.

Before you begin

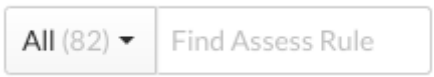
- Ensure that you have an Organization Administrator or Organization RulesAdmin role.
- Log in and select the correct organization.

Steps

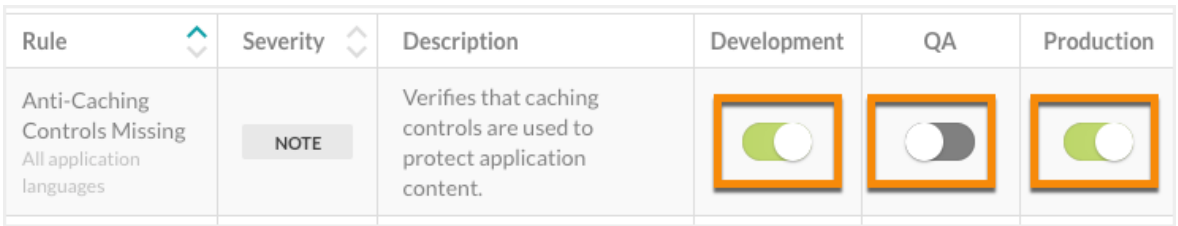
1. Under the user menu, select **Policy management**.
2. Select **Configure the default policy**.



3. Select **All** as the filter.



4. For each Assess rule, use the toggles to turn each rule on or off for each environment.



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.
- **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 and input 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.

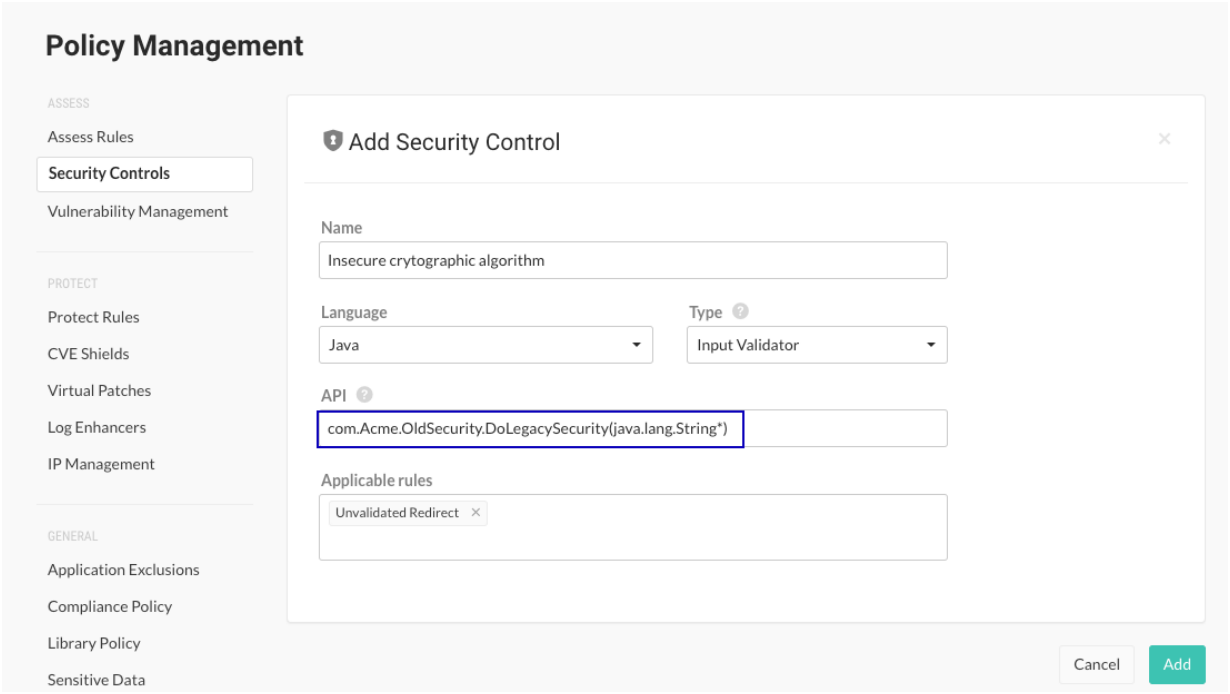
Supported languages

Security controls apply to Java, .NET Framework, and .NET Core languages 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*)
```



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

Security controls apply to Java, .NET Framework, and .NET Core languages 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 either one of these methods:
     - **Input validators** accept user input and take corrective action if unsafe data is received.
     - **Sanitizers** clean the data that is passed in, making it safe for consumption by any interpreter. Many sanitizers prevent one type of attack, but not another.
   - **API:** When specifying the API, consider these conventions:
     - Java must include 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 (*).
   - **Applicable vulnerability rules:** You can 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 Viewer 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.

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.

You can set in-app notifications (page 644) when vulnerabilities violate these policies. Administrators are notified of violations in-app and by email.

Auto-verification policies

Auto-verification (page 606) policies automatically change the status of a vulnerability that meets specific criteria to **Remediated - Auto-verified**:

- 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, Contrast views the two sessions as a single session.
  This action applies to vulnerability policies with a route-based trigger.
- 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.

An auto-verification policy can have a route-based or a time-based trigger .

Violation policies

Violation policies (page 608) 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.

Policy triggers

These trigger types activate a vulnerability policy:

- **Route**: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 certain amount of days.

Session metadata for route-based auto-verification

For optimal results from route-based vulnerability polices, add session metadata (page 447) 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 the best way to ensure that Contrast is tracking routes and vulnerabilities across an entire test run even you restart the agent and the application multiple times during the run.
  Contrast creates a unique session ID for every unique metadata-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.

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

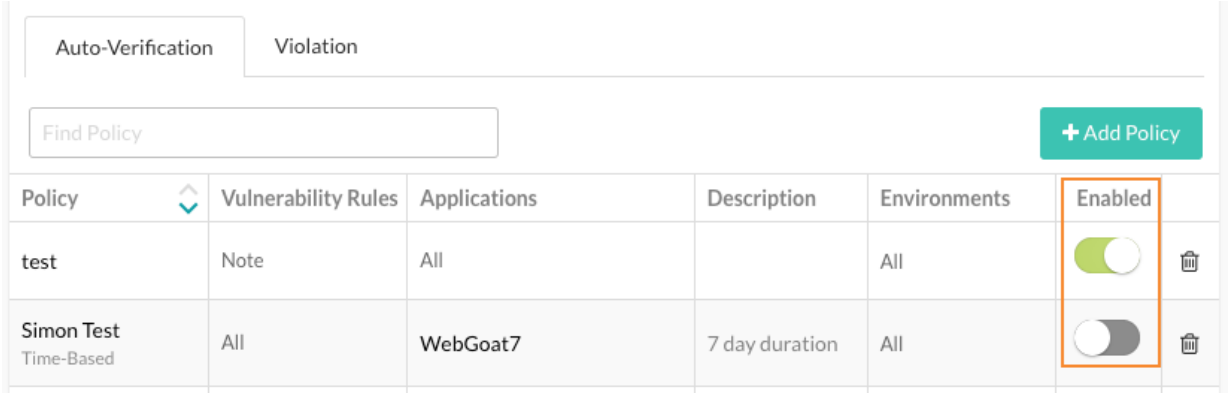
If multiple policies affect the same vulnerability, these rules determine how Contrast applies the policies:

- 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.

Set auto-verification vulnerability policy

Auto-verification policies for vulnerabilities (page 605) automatically change the status of a vulnerability that meets specific criteria to **Remediated - Auto-verified.**. These policies can be 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.



TIP

Use time-based auto-verification along with route-based auto-verification to find situations where routes change drastically from build to build. For example:

- Major code refactoring where you add new new routes and remove old routes.
- A route is no longer exercised because it is no longer valid.

Before you begin

- An Organization RulesAdmin or Organization Administrator role is required.
- 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
- Verify that you are using a supported framework. (page 608)

Steps

1. From the user menu, select **Policy management**.
2. Select **Vulnerability management**.
3. In Vulnerability policy, select the **Auto-verification** tab.
4. To add a policy:
   a. Select **Add policy**.
   b. In Name, enter a name for the policy.
   c. In Vulnerability rules, select one or more severity levels or Assess rules that you want to associate with the policy.
   d. In Applications, select one or more importance levels or applications that you want to associate with the policy.
   e. In Environments, select one or more server environment where the policy is applied: All environments, Development, QA, or Production.

f. Select the type of trigger you want to use for the policy (select one or both 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.
   - To set a route-based trigger, select **Auto-verify any existing vulnerability**, Route-based triggers only work for certain technologies with identifiable routes. A Route-based trigger takes precedence over a time-based trigger.

g. Select **Save**.

5. To update a policy:
   a. Select the policy.
   b. Update the values, as needed.
   c. Select **Update**.

Auto-verification supported frameworks

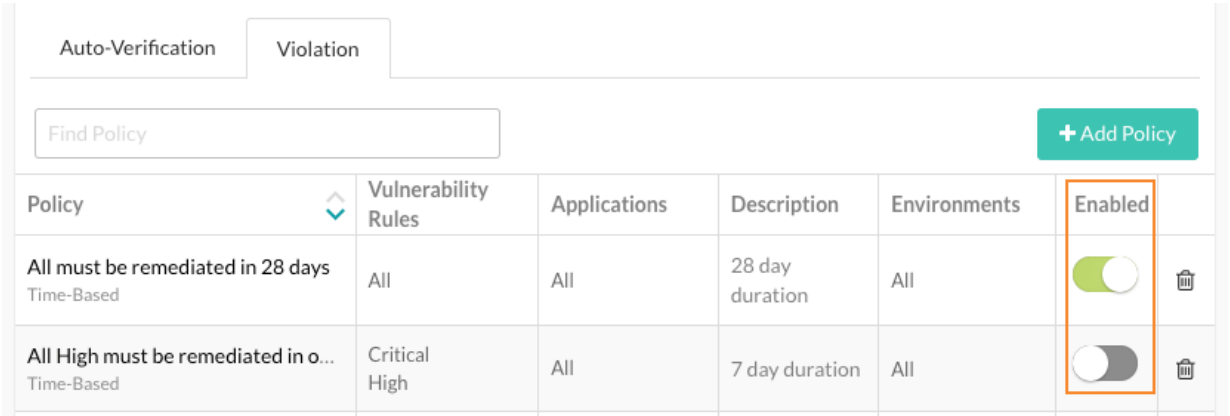
These frameworks support auto-verification policies:

- **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
- **Ruby**: Rails and Sinatra
- **Python:** Django, Pyramid and Flask

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.



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.
- **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 609) that monitor or block attacks in your application environments.

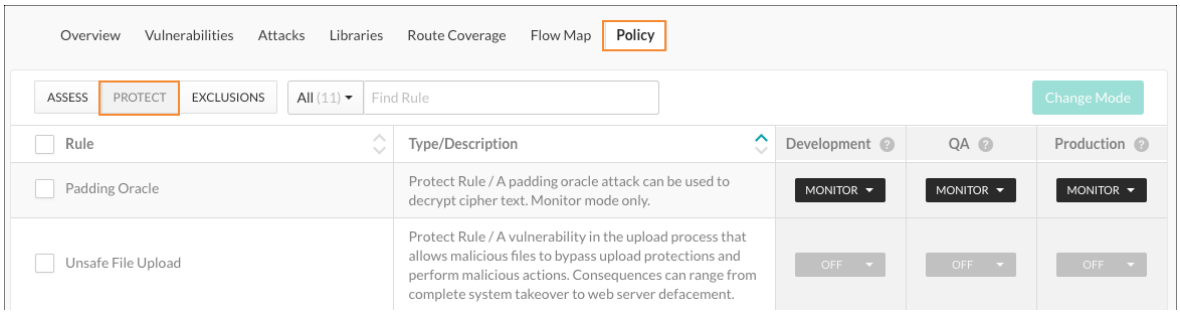
When you add new applications, Contrast applies a set of default Protect rules to them. You can change the modes for an organization's default Protect rules (page 612).

Before you begin

- An Application RulesAdmin role is required to set Protect rules for applications.
- An Organization RulesAdmin or Organization Administrator role is required to set Protect rules for an organization.
- Ensure that Contrast (hosted customers) or a SuperAdmin (on-premises customers) granted Protect permissions (page 707) for the organization.

Steps

1. Select **Applications** in the header.
2. Select an application name and select **Policy**.
3. Select **Protect**.



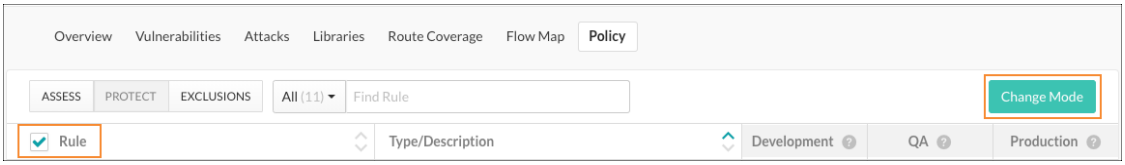
4. To find a specific rule, enter the rule name in the search box.
5. For each rule, set the mode for each environment:
   a. Select the dropdown for each environment.
   b. Select one of these modes:
      - **Off:** This mode disables the rule.
      - **Monitor:** The agent identifies and reports 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`.

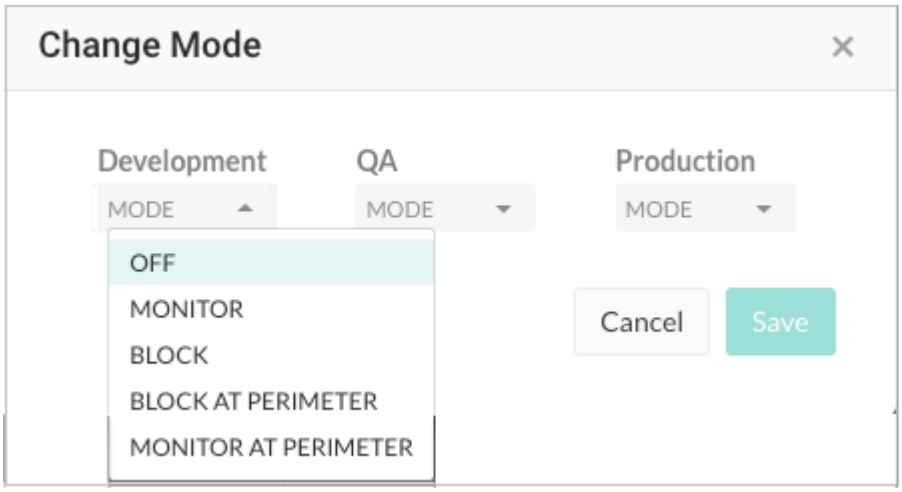
      - **Block at perimeter:** The agent blocks a possible attack before the application can process it. This option is not available for all rules.
      - **Monitor at perimeter:** The agent attempts to identify and report a possible attack before the application can process it. This option is not available 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.

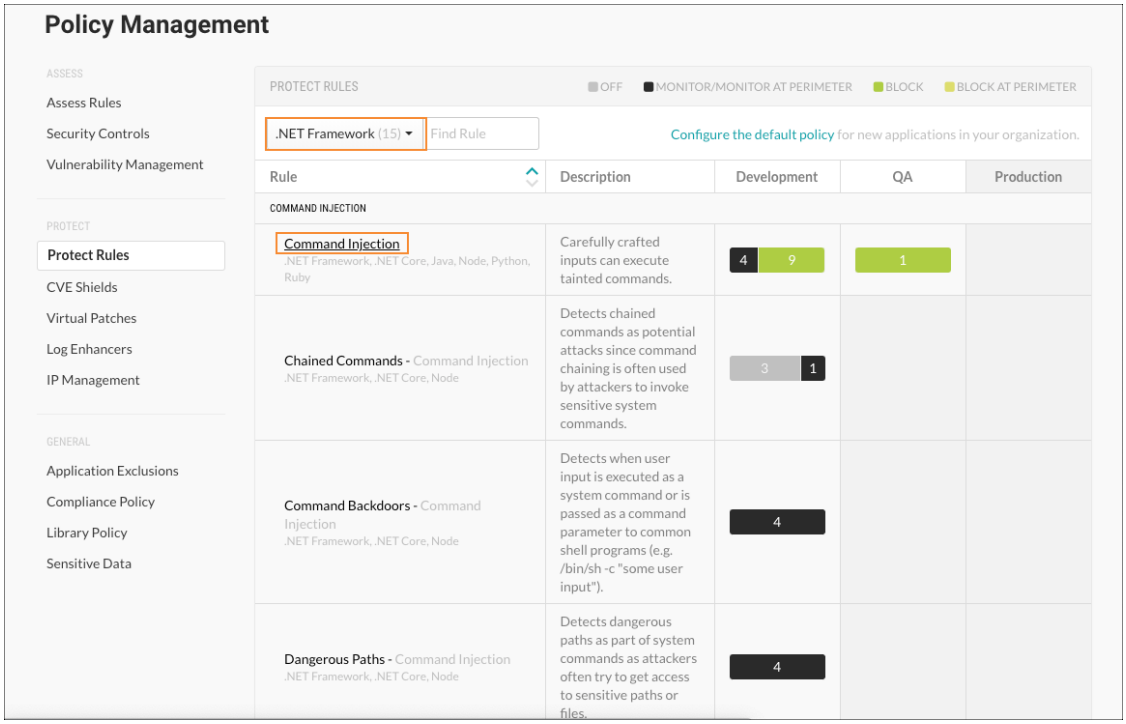
6. To apply settings to multiple rules, use one of these methods:
   a. Select the checkbox next to each rule that you want to change and and select **Change Mode**.
   b. To change settings for all rules, select the **Rules** checkbox and select **Change Mode**.



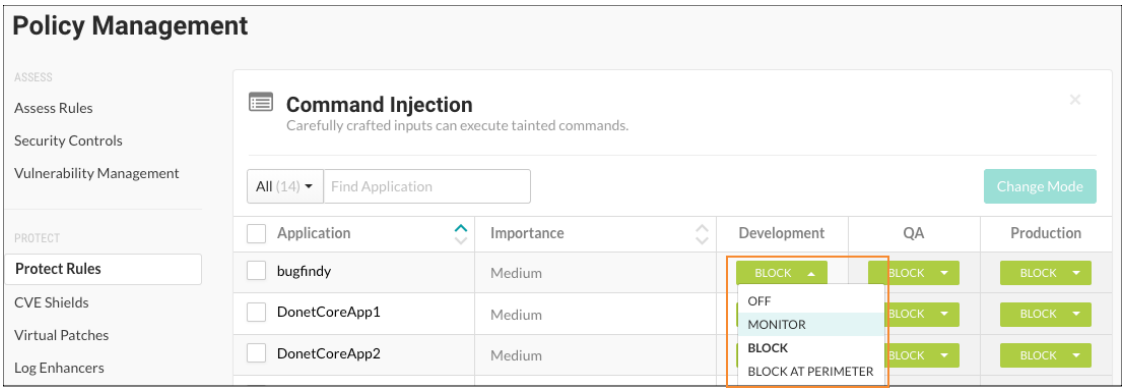
   c. In the Change Mode window, set the mode for each environment and select **Save**.



7. To set Protect rules for all applications in the organization that use a specific rule:
   This step requires an Organization RulesAdmin role.
   a. Select **user menu > Policy management > Protect rules.**
   b. 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.



   d. Use the dropdown to set the Protect mode for each environment.



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.

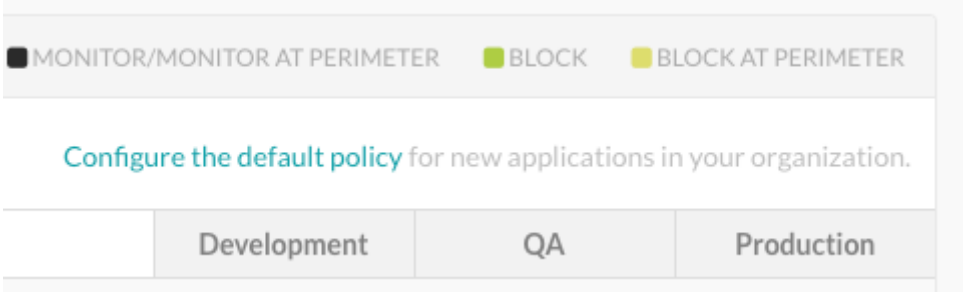
Use this procedure to change the default settings for Protect rules at an organization level. Changing these settings affects new application that you add to a Contrast organization. These changes have no affect on existing applications in the organization.

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 **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.

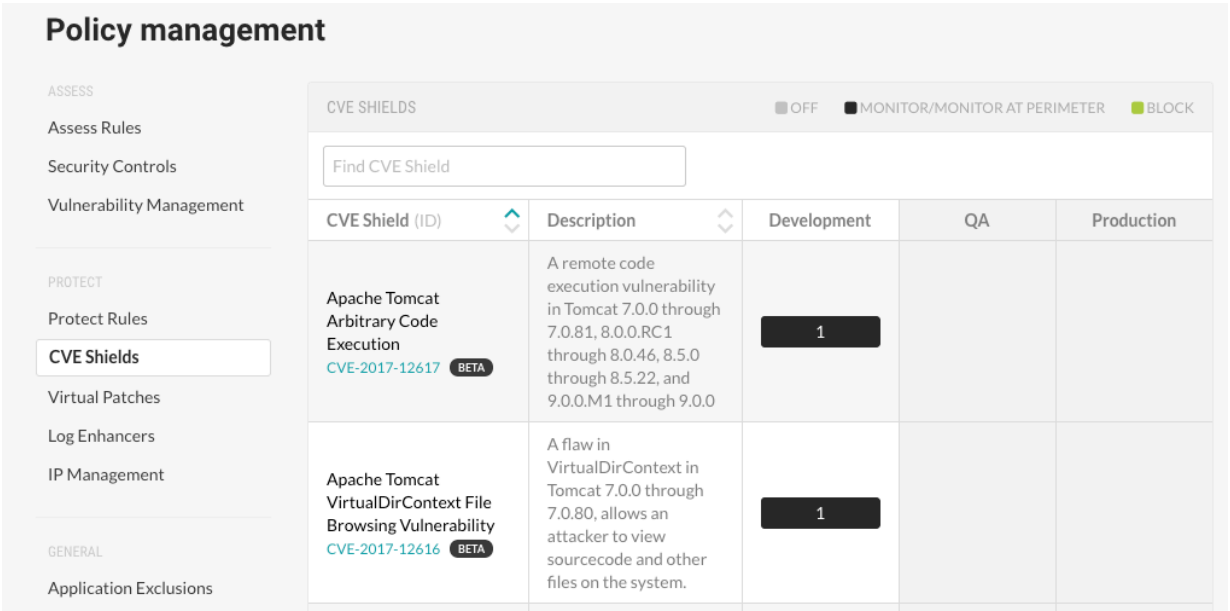
You only need CVE shields when the vulnerability isn't a common attack class like SQL injection or untrusted deserialization. In some cases, Contrast creates a CVE shield to get more data that is specific to a particular threat, even if there's an existing Protect rule that prevents the attack from occurring. 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 613)

Set modes for CVE shields (page 614)

View CVE shields

The CVE shields list displays the following information:



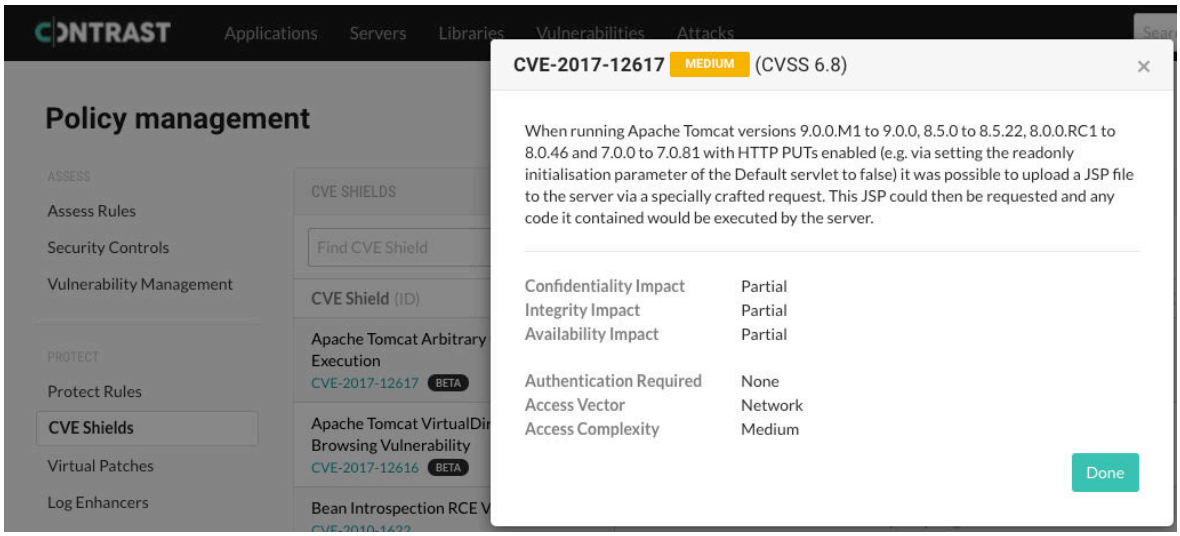
- The CVE shields that Contrast provides for specific CVEs.
- A description of the CVE.
- The environments in which the servers hosting an application are running.
- The mode configured for the CVE shield:
  - **Off:** This mode disables the CVE shield entirely.
  - **Monitor:** In this mode, the CVE 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.

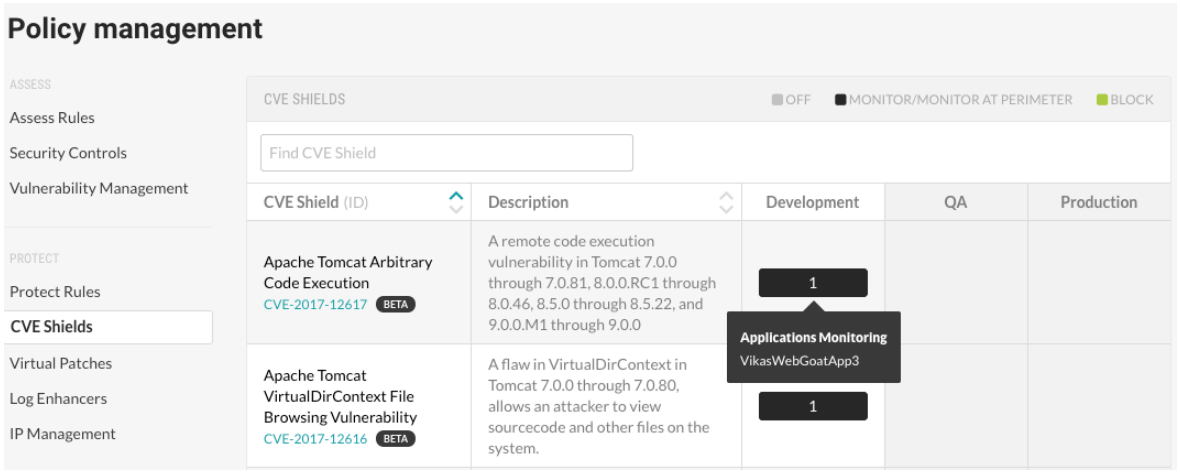
Steps

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.

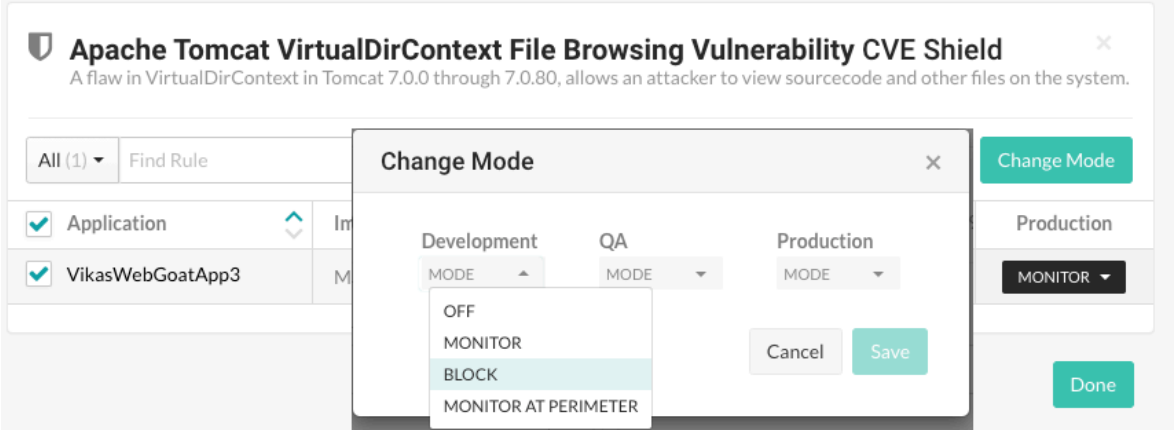
Before you begin

- **Required**: Check that you have Organization or Rules Admin permissions.
- Check that settings (page 484) for servers hosting your applications are configured to use the correct environments.

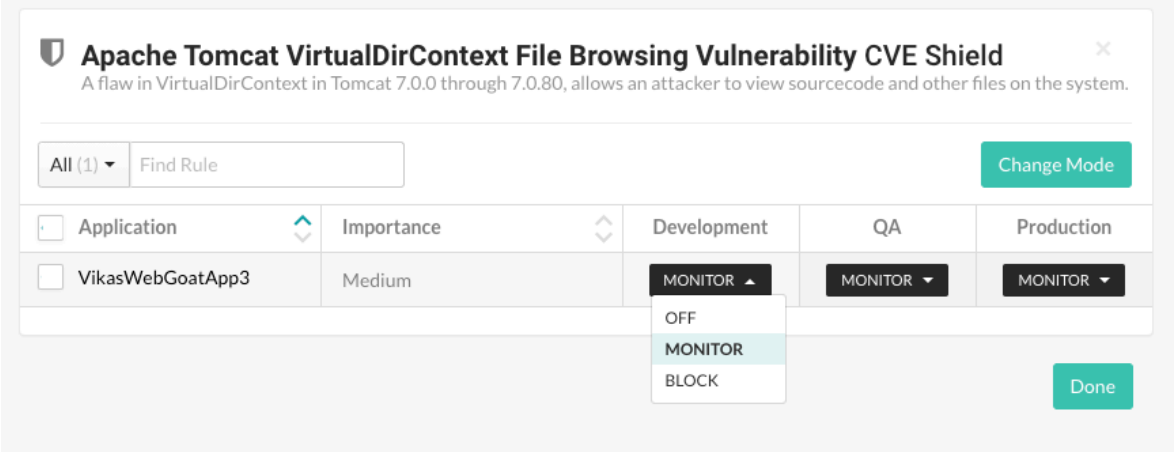
Steps

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:

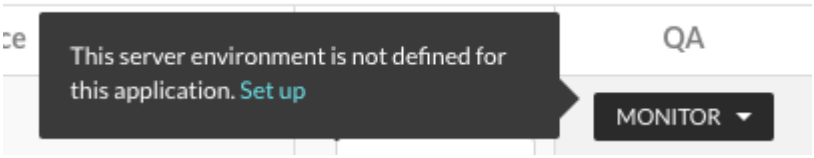


   a. To select all applications, select the **Application** checkbox. To select multiple applications, select the checkbox for each application.
   b. Click **Change mode**.
   c. In the Change Mode window, select the CVE shield mode for the selected applications in a one or more environments.
   d. Click **Done**.
4. To set the mode for a single application:



   a. 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.



b. Select a CVE shield mode for the application in the selected environment.
c. 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 affect on existing applications in the organization.

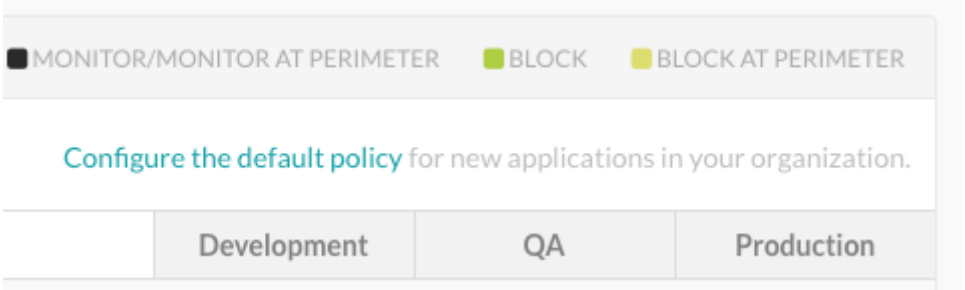
Before you begin

- 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.

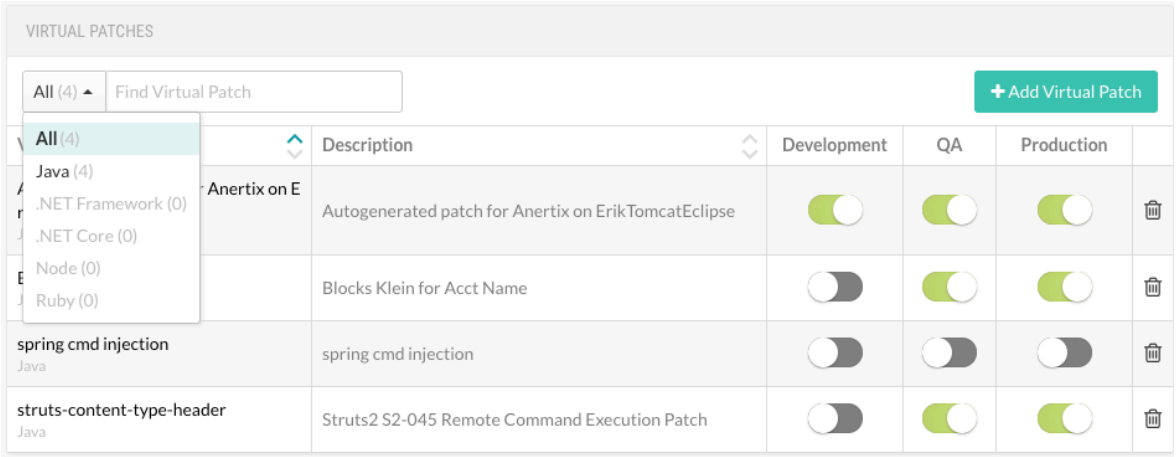
Manage virtual patches

Virtual patches are custom, short-term rules that block HTTP requests matching specific criteria (for example, URL, parameter keys or values, etc.) before an application can process them.

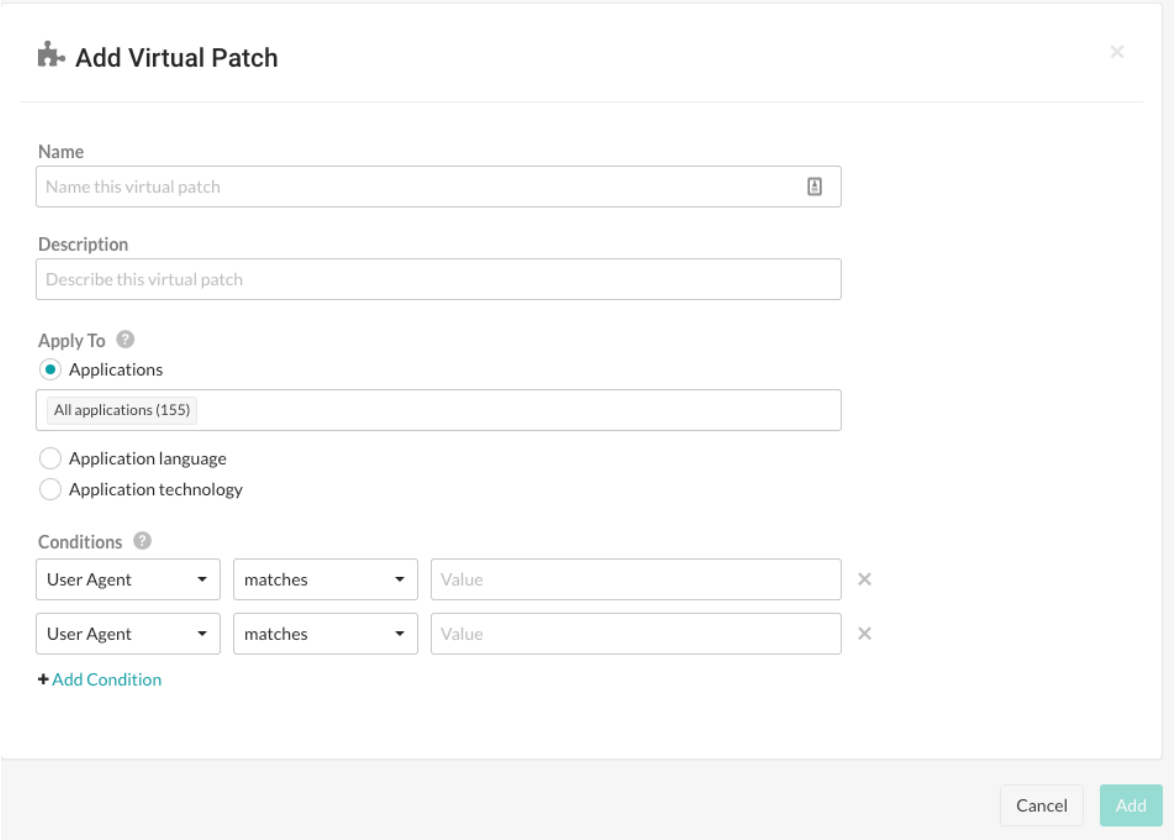
Organization Administrators and RulesAdmins can view and manage virtual patches.

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.



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** .



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 application(s). Select **Add another condition** in a separate row, if necessary.
7. Select **Add** to save the configuration.

Add or edit log enhancers

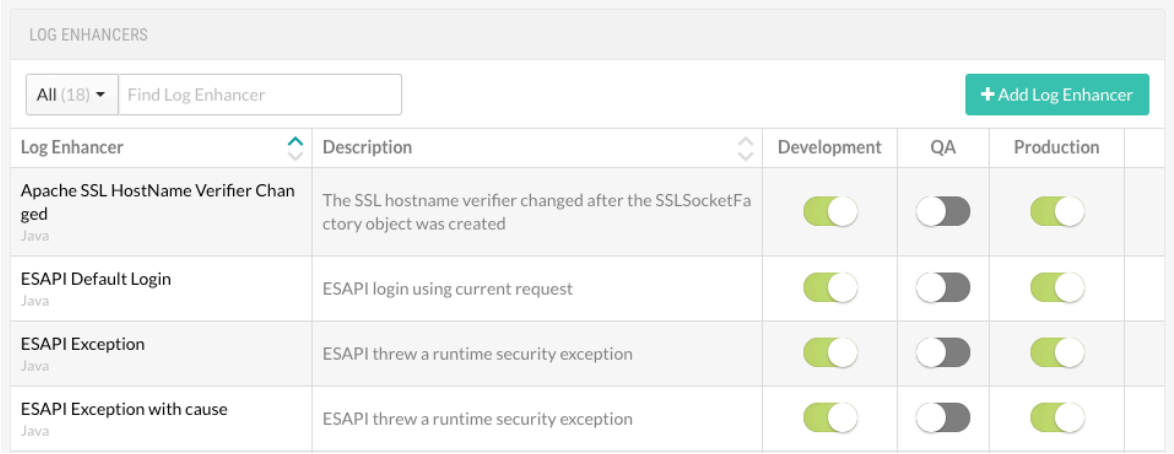
Log enhancers are instrumentation instructions that allow the Contrast agent to log additional parameters and data in the application, without requiring any source code changes.

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.

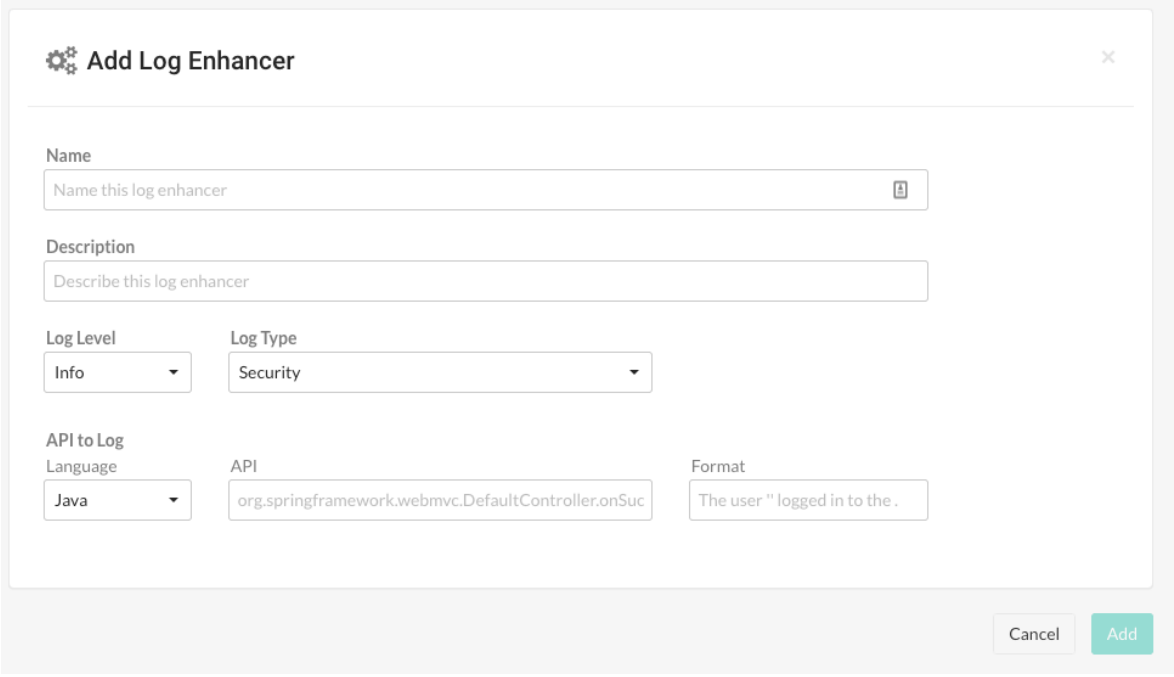
NOTE
Starting in August 2021, new organizations include an optimized set of log enhancers. This configuration is designed to provide the highest value to users, including enhanced performance.

To add, edit or delete a log enhancer:

1. Under policy management (page 600), 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.



3. In the window that appears, enter a **Name** and **Description**.



4. Enter a **Log level (page 749)** and **Log type**.
5. 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.
6. 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, or Ruby agents, you can add an application exclusion (page 620) under policy management, or from the list of attack events.

You can view a list of existing exclusions either at **Applications > Your application name > Policy > Exclusions** or in the **user menu > Policy management > Application exclusions.**

Add application exclusions

Java, .NET Framework, .NET Core, Node.js, Python, and Ruby agents let you use an application exclusion (page 620) to exclude certain applications, or parts of them, from security analysis.

Currently, Go and PHP agents do not support application exclusions.

NOTE

Message queue exclusions are available for Java agents only. This feature is currently not generally available. To get access to Message queue exclusions, contact Contrast support.

Before you begin

- An Admin or RulesAdmin role is required.
- Node.js, Python, and Ruby agents do not support code exclusions.
- Only Java agents support 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 s 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 Examle: somefile\.jsp
```

   Use these regex examples (page 622) 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.
     - 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 where they've come from.
     - **These URLs:** Specify a set of paths to which to apply the exclusion. You can use regex (page 622) and wildcard expressions. (page 622)

     > **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 622) and wildcard expressions (page 622).
   - **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 622) and wildcard expressions (page 622).
6. 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.

Regular expression reference

Use this table, and the examples below, for reference when creating application exclusions (page 620):

Effect	Pattern	Example pattern	Example match
Start of a string	`^`	`^w+`	`Start` of a string
End of a string	`$`	`w+$`	End of a `string`
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
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

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	`/myapp/.+`	Excludes all paths with the initial URL of `/myapp/`
Exclude one character from subpath	`/.yapp`	Excludes all subpaths that are 5 characters and end in `yapp` (like `myapp`)
Exclude one subpath explicitly	`/myapp/thispath`	Excludes only `/myapp/thispath`
Exclude path ending	`/.*ignore`	Excludes all paths ending in `ignore`
Exclude paths containing	`/.*value.*`	Excludes all paths containing `value`
Exclude path containing	`/.?value.*`	Excludes all paths either starting with `value` or paths that have one character before `value`
Exclude paths where a period (dot) is used as a literal character and not a wildcard.	`/myapp\.js`	Excludes only `myapp.js` 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 600), 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 624)
- Manage source names (page 625)

Deny or allow IP addresses

Use IP denylists or IP allowlists to manage IP addresses (page 624) in your organization.

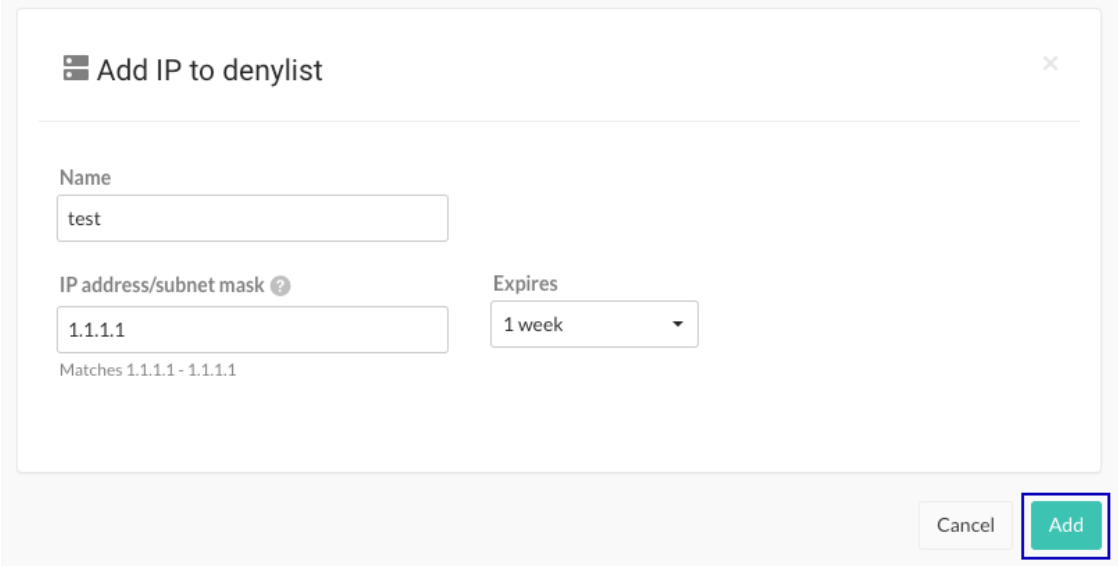
Before you begin

- 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.

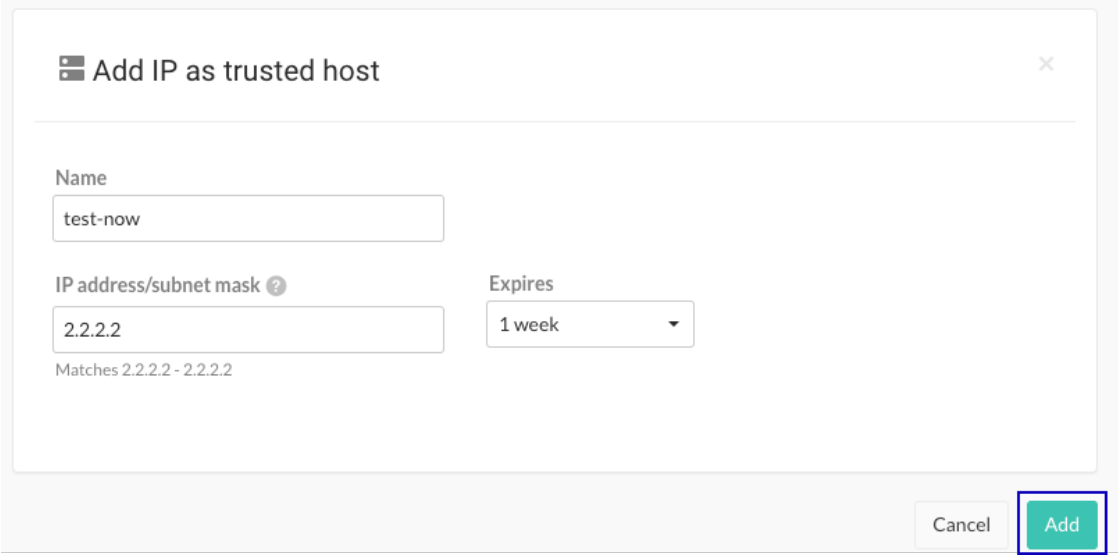
Steps

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**.



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 542).
   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 646).

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 628).

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 627) 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. Check the box next to **Mask entire body** to enable redaction of the entire HTTP request body. This will apply to all applications in your organization.
4. 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.
5. Click on the name of the data type in the grid to add custom keywords. In the **Edit sensitive data type** form, select **Add custom keywords** to add more keywords and specify the applications to which they apply. **Default keywords** aren't editable, and apply to all applications.
6. 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 632), 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 600) or Editor (page 434) can do, plus, at an organization level you can:

- Configure organization settings (page 632)
- Enable Assess (page 629)
- Enable Protect (page 629)

NOTE
Organization Administrators have the highest organization level permissions. Other organization roles (page 744) also have capabilities that span across their organization.

If configured at a system level (page 705), users can fill the Organization Administrator role across multiple organizations.

Enable Assess

To enable Assess:

1. Log in to Contrast as an Organization or Application Administrator.
2. Select **Servers** in the header.
3. Either scroll or use the search at the top of the page to find the server(s) you want to analyze. Turn the Assess toggle on (green). There are three ways to do this:
   - Select the toggle in the Assess column of the list of servers.
   - Select the server name to view information about that Server, and use the toggle there.
   - Under organization settings, select **Server** and use the Assess toggle there.
4. Enabling Assess requires an Assess (application) license. To license an application, select **Applications** in the header. Select **Unlicensed** next to the application name and you'll be prompted to apply a license.

   NOTE
   Under organization settings (page 632), with Organization selected in the left navigation, Organization Administrators can select the box next to **Automatically apply licenses to new applications** so this doesn't have to be done manually for every application.

5. 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
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 have a license.

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, restarting the applications is required for Protect to 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 707)
    These roles can also configure which user roles have access to Protect data.
- Ensure that you have licenses (page 633) that you can apply to servers.
- To enable Protect for users, an Organization Administrator role is required.

Steps

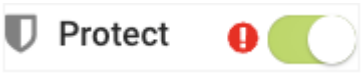
1. To configure the Protect settings in the Contrast web interface, log in to Contrast.
2. Enable users to see and use Protect data:
   a. In the user menu, select **Organization Settings**.
   b. Select **Users**.
   c. For each user who needs access to Protect data, turn on the Protect setting ( )
   d. To have the new setting take effect, tell users to log out of the Contrast web interface and log in again.
3. Enable Protect on servers:

   NOTE
   To automatically allocate licenses (page 633), 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.

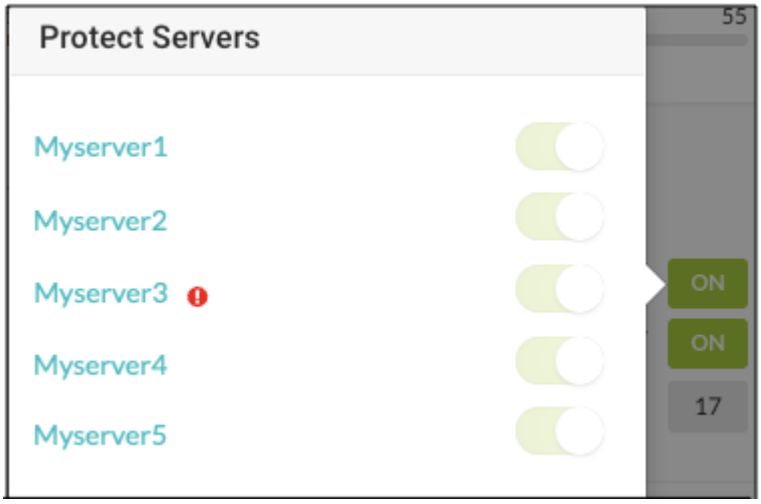
   a. Select **Servers** in the header.
   b. Select a server.
   c. To enable Protect in the Contrast web interface, turn on the Protect setting ( ). Use any of the these methods:
      - Turn on the setting in the **Protect** column of the list of servers.
      - Select the server name and in the Overview tab, turn on the Protect setting.
      - Under Organization settings, select **Server** (page 640), select an environment, and turn on the Protect setting.

      IMPORTANT
      The Protect setting in an agent configuration file overrides the setting from the Contrast web interface.
4. To verify that Protect is turned on for a specific server, in the Servers tab, select the server and then select **Overview**.

If one or more applications that the server is hosting are not configured to use Protect, a warning icon displays next to the Protect setting.



5. To determine if an application is using Protect on each server hosting it, go to the Applications page:
   a. Select **Applications** in the header.
   b. Select an application.
   c. In the Overview tab, under each environment, check that the Protect status is **On**.
      The number of servers hosting the application that have Protect turned on displays next to the Protect status in the format of **x of y** (for example, **11 of 17)**
   d. To see if the application is configured to use Protect for each server hosting it, select the **Protect** status (ON).
      If the application is not configured to use Protect on a specific server, a warning icon displays next to the server name.



6. (Optional) Configure an application to use Protect:
   a. Update the Protect setting in the agent's configuration file (page 30) to `true`.
   b. Restart your application to have Protect take effect.
      Once the application restarts, Contrast starts monitoring attacks, blocking attacks, and displaying attack data in the Contrast web interface.

> **IMPORTANT**
> The Protect setting in the agent configuration file overrides the settings from the Contrast web interface.

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 are turned on for an organization.
- 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:

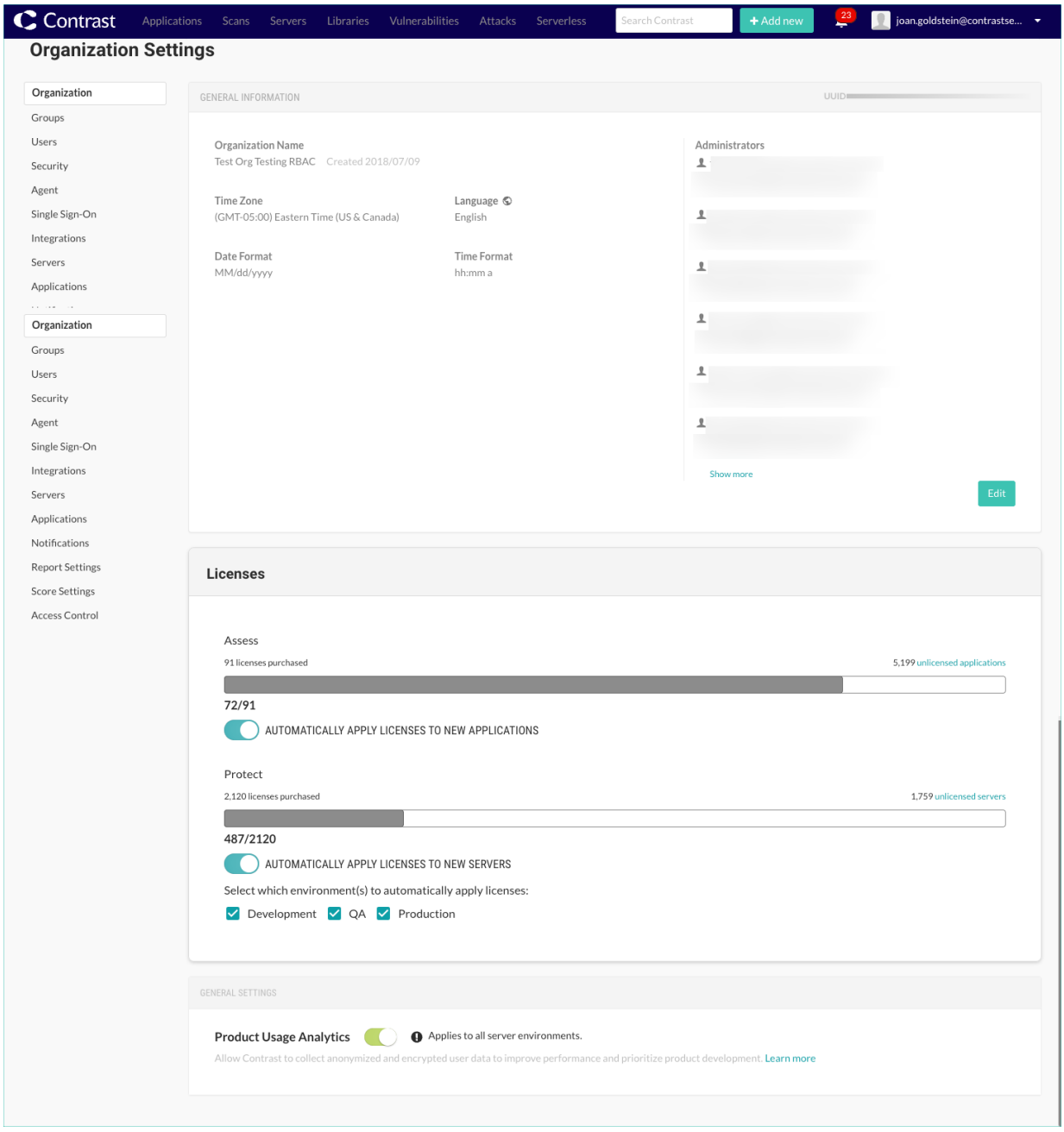
- **Protect is turned on in the agent configuration file**

- 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 organization(s):



1. Log in to Contrast as Organization Administrator.
2. If you have multiple organizations, select the name of the organization you want to configure in the **user menu**.
3. Select **Organization settings** in the **user menu**. Here you can access settings for your:
   - Organization (general information (page 633), license (page 633), diagnostics (page 638))
   - Users, groups and permissions (page 635)
   - Security (passwords (page 639), two-step authentication (page 639), IP range (page 639), email domain restrictions (page 639))
   - API (view organization and agent keys)
   - Single sign-on (SSO) (page 640)
   - Integrations (page 553)
   - Servers (page 640)
   - Applications (page 642)
   - Notifications (page 643)
   - Report Settings (page 645)
   - Score Settings (page 646)

Configure general organization information

1. Under organization settings (page 632), 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 703).)
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), English, or Spanish

   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**.

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 730).

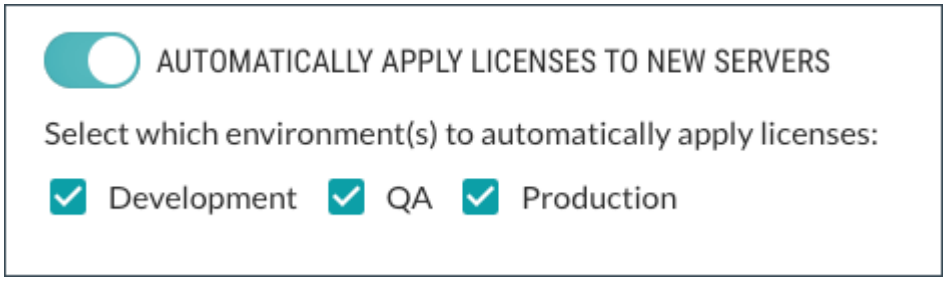
Steps

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.



   The Protect settings in an agent configuration file override this setting for the server hosting that application.
   For example, if you turn on this setting and the setting for Protect in an agent configuration file is set to `false`, Contrast turns off Protect for the server hosting that application. No Protect licenses are applied to the server.

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 444).
   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.

   c. Restart applications that the server is hosting.

IMPORTANT
The Protect setting in an agent configuration file overrides this setting.

Protect 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 are turned on for an organization.
- 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 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 is turned off for the server, based on an organization setting or you turn off Protect after the server is registered with Contrast, no license is applied.
  - If Protect is turned on for the server, based on an organization setting or you turned on Protect after the server is registered with Contrast, a license is applied, however Protect is turned off in the agent.

Manage users, groups and permissions at an organization level

If allowed at a system level (page 702), an Organization Administrator can manage user permissions for the organization by:

- Adding a user or editing user settings (page 635)
- Managing organization permissions with access groups (page 636)

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 must Contrast installations, the default (page 702) 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 702).

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 744) that apply to all applications in this organization or create a custom access group (page 636).
   - **Application access groups:** Select one of the default application roles (page 743) 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 707).)

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 742).

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 632), 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 705) 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.

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.

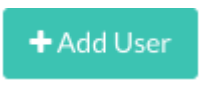
Before you begin

- An Organization Admin role is required.
- API Only users can access Contrast's REST API but cannot log in to the Contrast web interface.

Steps

To create an API only user:

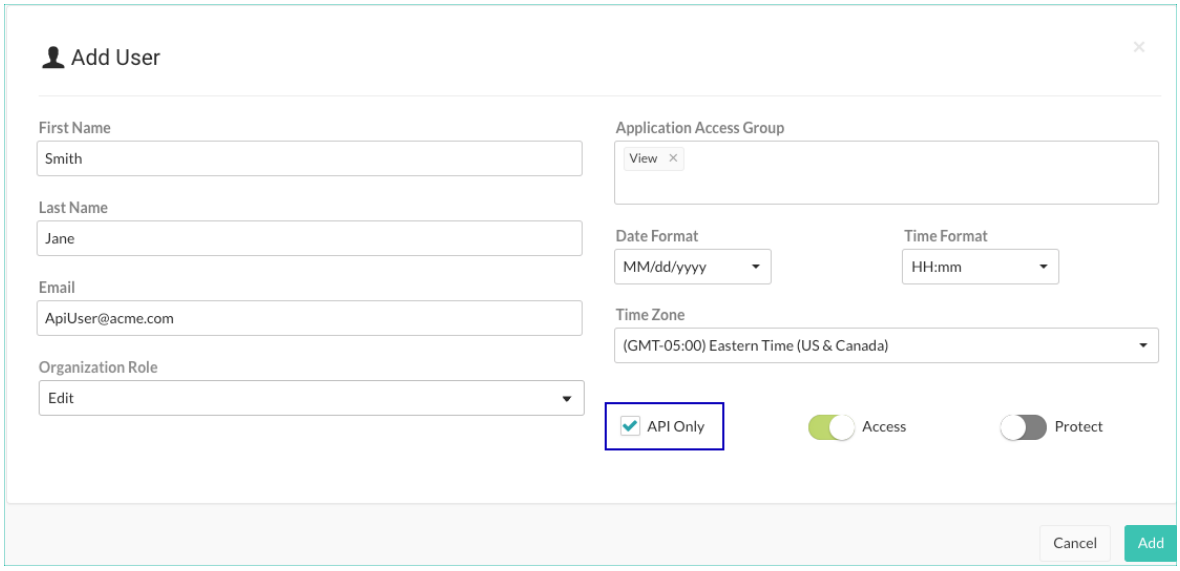
1. From the user menu, select **Organization settings**.
2. Select **Users**.
3. Select **Add User**.

4. Enter the user name , email address, and time zone information.
5. Select the **Role**.
   Best practice: Select **Edit** (page 744) 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.

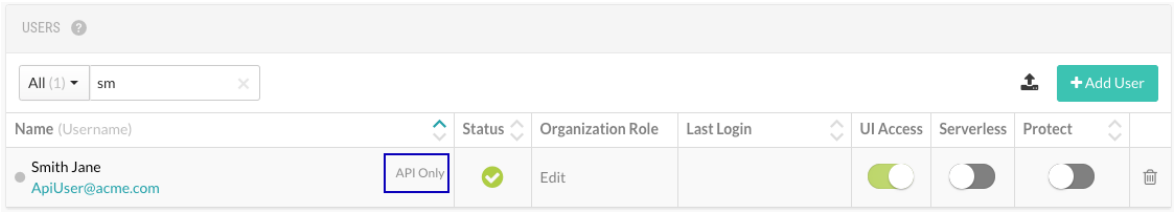
Best practice: Select **View** (page 743) or **Edit** (page 743) for the Application access group. Depending on the API endpoints you want to call, and if you are trying to GET (read) or POST (write), the API only user might require the higher Edit permission instead of View.

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.



9. If you are using access groups to restrict access to specific applications, add the API only user to the groups (page 636) 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.
    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 632), 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 729).

Set a password policy at an organization level

As long as it is allowed at a system level (page 725), as an Organization Administrator, you can regulate passwords within your organization by creating a password policy.

1. Under organization settings (page 632), 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 lockout**.
   - 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 days 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 632), 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 two-step authentication at an organization level

To enable or disable two-step authentication:

1. Under organization settings (page 632), select **Security** in the left navigation.
2. Turn the toggle on (green) to enable two-step authentication.
3. Two-step verification is enabled and users can choose how they want to receive two-step authentication notices. (page 436)

NOTE
For on-premises customers, this must be allowed at a system level (page 713).

If a user belongs to multiple organizations, their default organization determines their two-step authentication settings.

Configure single sign-on (SSO) at an organization level

For on-premises customers, single sign-on can be configured at a system level (page 722). For hosted customers, Contrast Security configures authentication; however, an Organization Administrator 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.

1. Under organization settings, select **Single-sign-on** and click the link to **Get started**.
2. You may receive a warning window regarding the implications of changing authentication. Please read it carefully before proceeding.
3. Use the provided information to set up Contrast with your IdP.
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, check the box 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, contrastsecurity.com)
6. Click **Test connection** button to test the configuration. If an error occurs, Contrast provides a debug log for troubleshooting. (This test only validates the metadata and Contrast's ability to connect to the IdP.)
7. Once the test is successful, select **Save.**
8. 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 and then 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.

Before you begin

- An Organization Administrator role is required.

Steps

1. Under organization settings (page 632), select **Servers**.
2. Use the dropdown to choose the environment in which you want to apply the default (development, test 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 691) 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 stacktraces 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 33) 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 chose 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 the **Retain Library Data** toggle. 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 the **Agent diagnostics** toggle. Contrast uses this data to improve rules, performance, and to prioritize product improvements.

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 632), 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 623) 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 Session metadata, to set the default session metadata filtering (page 445) to the most recent session, select **Filter application details by most recent session**.
   This selected filter affects the applicaition's Vulnerabilities and Route coverage tabs.
5. 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.
6. Use the **Custom Fields** section to configure custom metadata (page 642) that should be provided for each of the applications in your organization.
7. Select **Save**.

Create custom fields to request application metadata

You can configure requests for custom metadata that is collected whenever you add a new application to Contrast.

When you install and configure an agent (page 30), 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 custom 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 in the custom fields are required for Agent configuration and YAML file download. The downloads will be disabled without the data.

Before you begin

- An Organization Administrator role is required.

Steps

1. Under organization settings, select **Applications**.
2. Under Custom fields, 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 custom field 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 644) for Contrast notifications for all users in their organization. Individual users can choose how they want to receive these notifications (page 437).

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 733) 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 662) 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 633) 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 605)
- **Library policy violation:** A library is in violation of an existing library policy (page 626).
- **Compliance policy violation:** An application is in violation of an existing compliance policy (page 623).

Administrator and RulesAdmin users at the application and the organization level must receive policy violation notifications. You can manage your notifications (page 437) 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 662) 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 437).

To configure default notification settings for an organization:

1. Under organization settings (page 632), 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 or "Library"; the default selection is "All".
   - **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 has been onboarded. Click in the field to set this notification for "Application" or "Server"; the default selection is "All".

- **Email digest:** A daily summary of Contrast activities. (Email only)

> **NOTE**
> 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.
5. Click in the multiselect field to choose the **Applications** for which this notification applies.
6. Choose the **Application Tags** for which this notification applies.
7. Choose which organization **Users** should receive the notifications.
8. 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.

Set report defaults

Report settings offer a single interface for Organization Administrators to define the template of hard-copy compliance reports (page 548).

1. Under organization settings (page 632), select **Report Settings**.

2. Define the default values for reports created within the organization:
   - **Report type:** Click to select a report type.
   - **Vulnerability status:** Confirmed, Not a problem, Remediated, Reported, Fixed
   - **Vulnerability tag:** This will apply to any vulnerability with the designated tag.
   - **Inclusions:** Select the box to include status of vulnerabilities or notes on the vulnerabilities in the report.
   - **Custom footer:** Add a custom footer you want to appear in the reports.
3. You can run a report to see the results. The defaults will pre-populate the report generation window, but the user can still make any necessary changes.

Customize score settings at an organization level

Contrasts designates an application score (page 747), which can optionally depend on a library score (page 496). To customize score settings at an organization level:

1. Under organization settings (page 632), select **Score settings**.
2. 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.
3. 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.
4. 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.

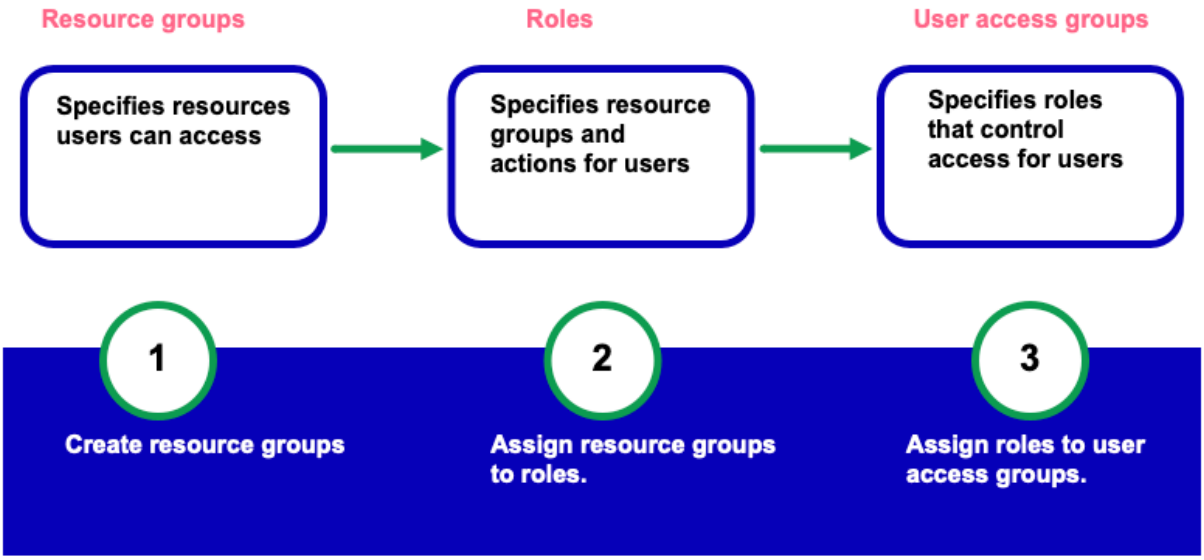
Role-based access control for Contrast Scan

To give users access to Contrast Scan, set up resource groups, roles, and user access groups.

NOTE

In the Contrast web interface, if you don't see the Access Control setting under Organization settings. contact Contrast support.

To manage access for all other Contrast capabilities, set up organization users and access groups (page 635).



Access control settings

Access control for Scan requires these settings:

- **Resource groups: (page 648)** These groups determine which Scan projects users can access, depending on their assigned role.
  You can use built-in resource groups or create custom groups.
- **Roles: (page 652)** Roles define the Scan actions that users have permissions to perform.
  You can use built-in roles or create custom roles.
  The built-in RBAC Admin role automatically includes all users with an Organization Admin role. This built-in role lets you manage all user access groups, roles, and resource groups for Contrast Scan.
- **User access groups: (page 657)** 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 an visual method for setting up and managing access control. This method requires an 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 elements. Doing so will help make managing a large number of custom elements 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.

Requirements for all custom elements are:

- Names can contain up to 50 characters.
- Descriptions can contain up to 1024 characters.
- You can use special characters and spaces in names and descriptions.

Resource groups for Contrast Scan

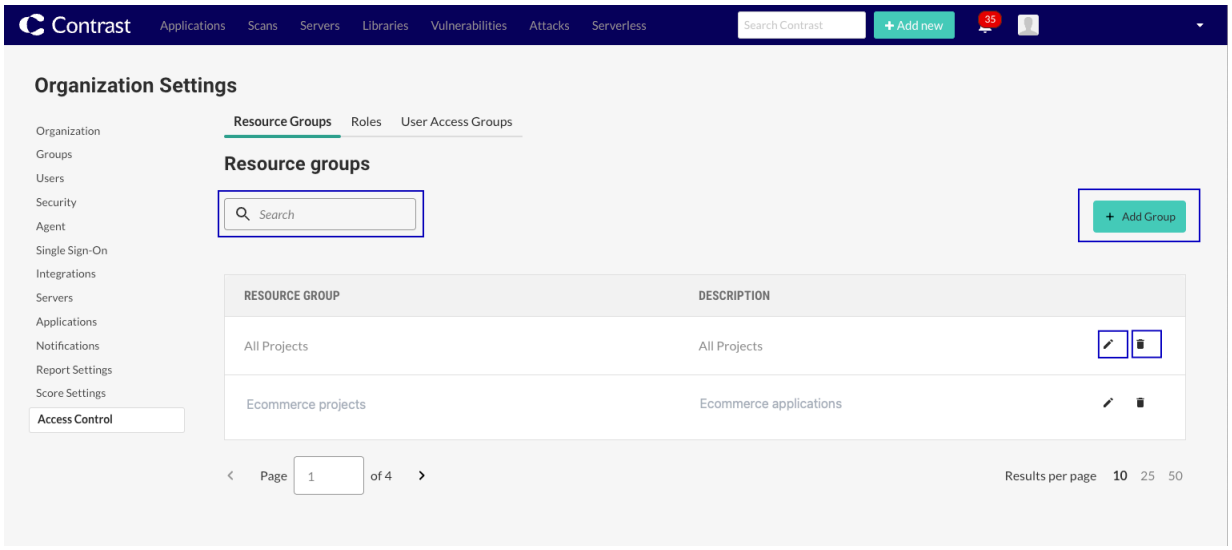
Resource groups let you specify the Contrast Scan capabilities that users can access, based on their assigned roles.

NOTE
Currently, only scan projects are supported for resource groups.

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 for Scan (page 649).
  - Edit a resource group for Scan (page 650).
  - Delete a resource group for Scan (page 651).



Built-in resource group for Contrast Scan

You can select this built-in resource group for Contrast Scan roles:

- **All projects:** includes all scan projects in the organization

NOTE
You cannot change the settings for a built-in resource group.

Add resource groups for Contrast Scan

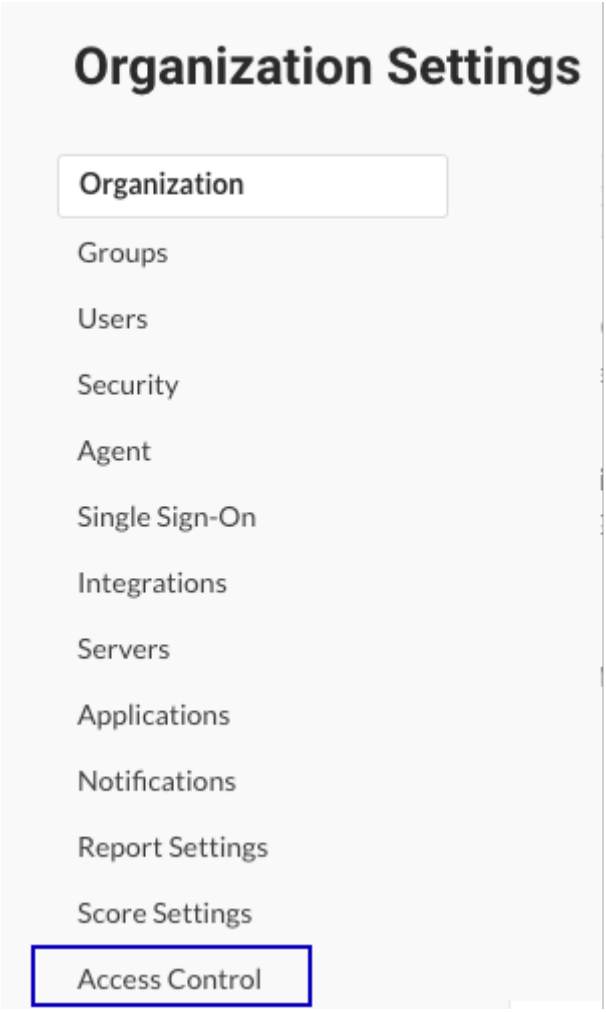
Resource groups let you specify the Scan components that users can access. Currently, scan projects are the only supported resources.

Before you begin

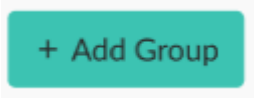
- An Organization Admin role is required.

Steps

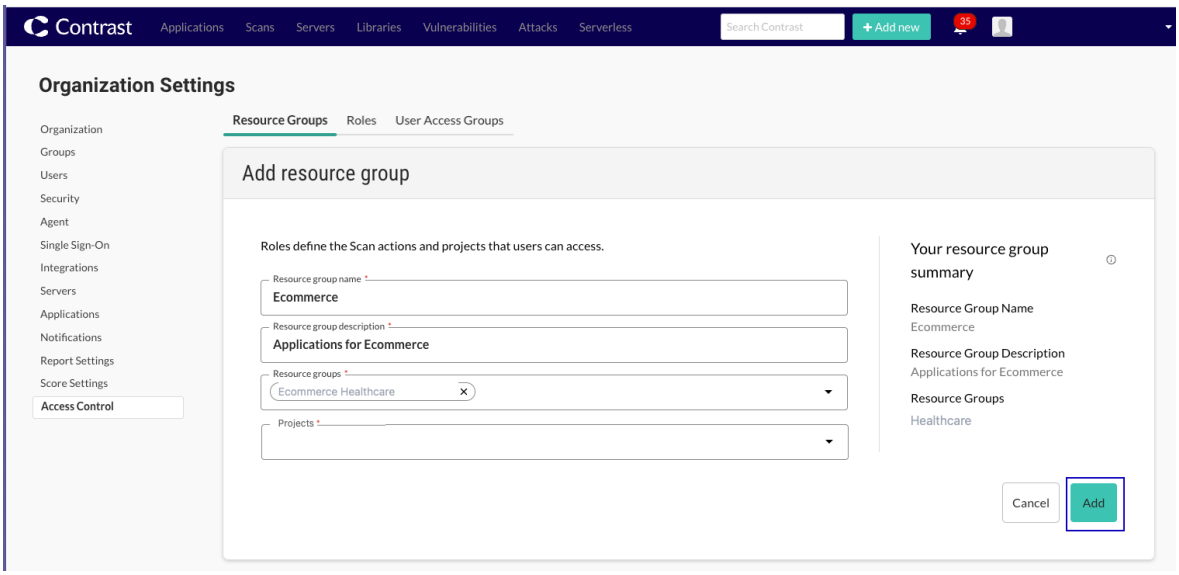
1. From the user menu, select **Organization settings**.
2. Select **Access control**.



3. Select the **Resource groups** tab.
4. Select **Add group**.



5. In the Add resource group for Scan window, specify these settings:



- **Resource group name:** Specify a descriptive name.
  The name must be unique for the organization. Use up to 50 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 1024 characters, including spaces and special characters.
- **Resource groups:** Select resource groups to include in this group.
  If you select resource groups, selecting an individual project is optional.
- **Projects:** Select individual Scan projects to include in this group.
  If you select projects, selecting a resource group is optional.

6. Select **Add**.
   You can now assign this resource group to a role.

Edit resource groups for Contrast Scan

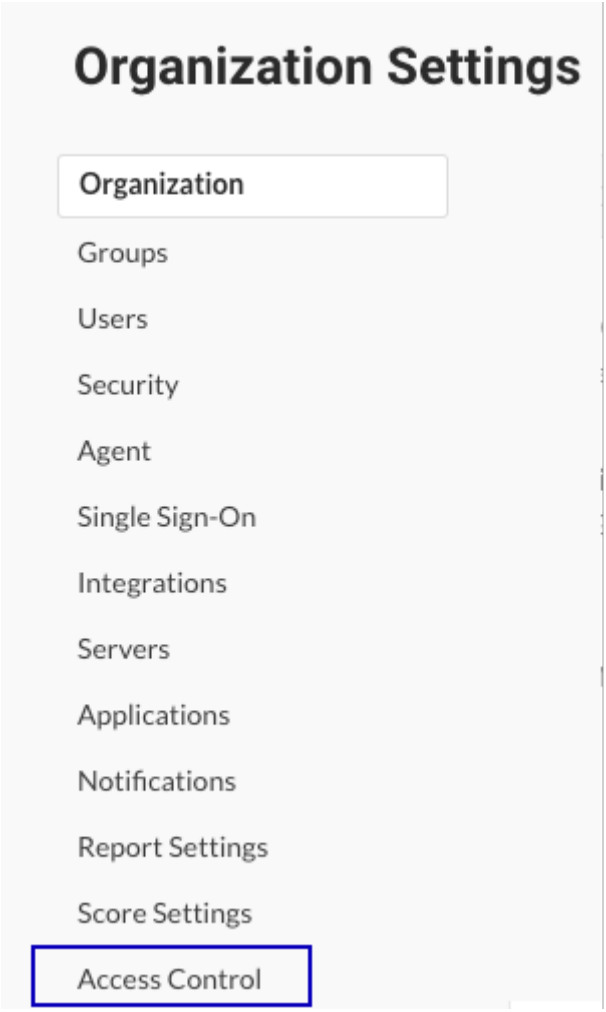
Use this procedure to change settings for a resource group. You can change any or all of the settings.

Before you begin

- An Organization Admin role is required.

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 resource groups for Contrast Scan

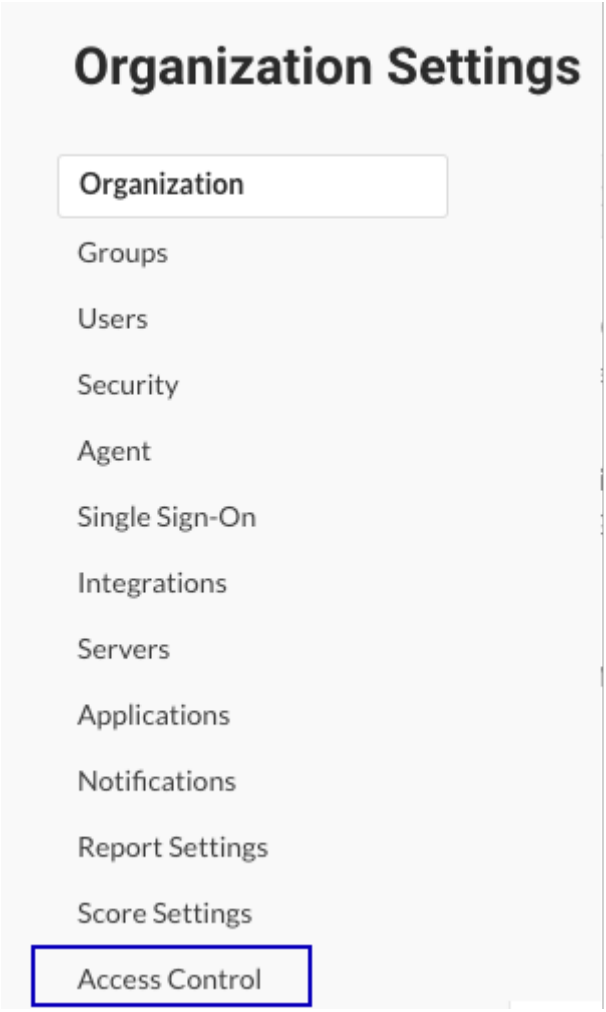
Use this procedure to delete a resource group.

Before you begin

- An Organization Admin role is required.

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**.

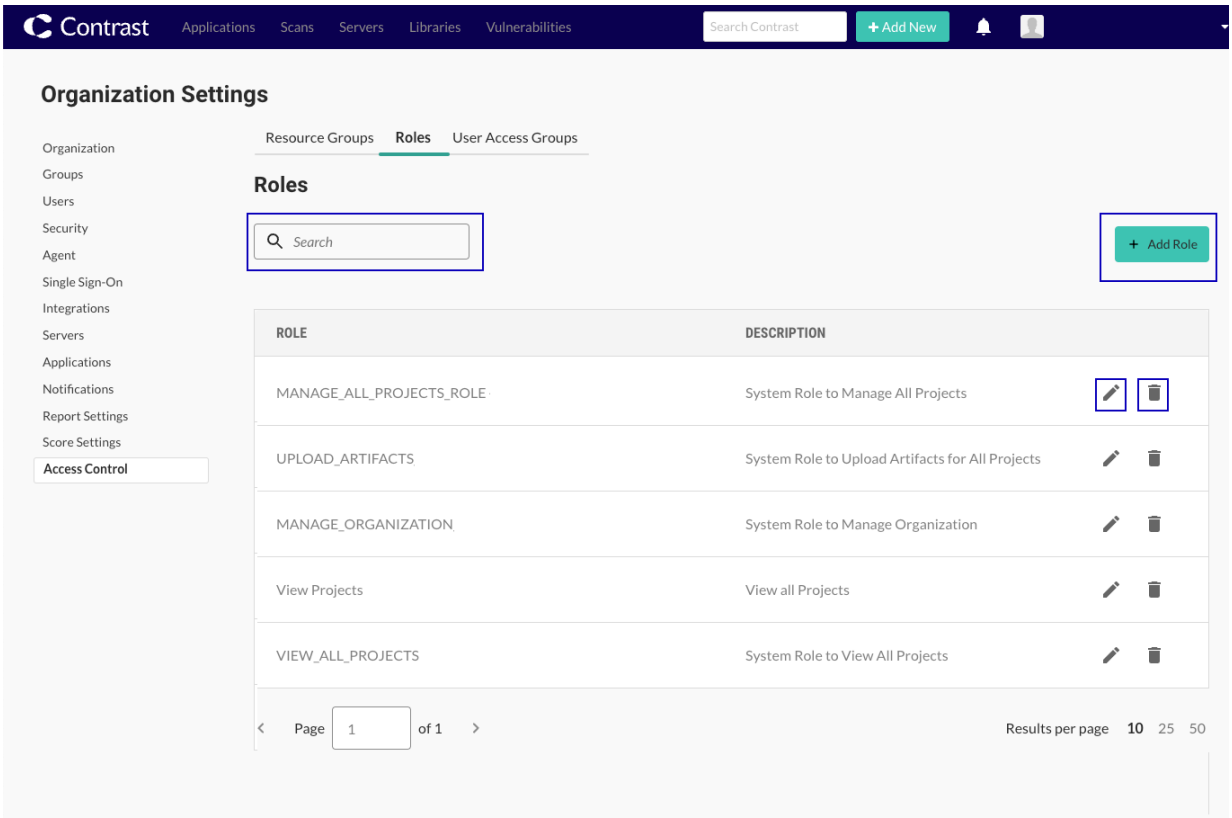
Roles for Contrast Scan

Roles let you define the Contrast Scan capabilities that users with a specific role can access.

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 for Scan. (page 653)
- Edit a custom role for Scan (page 655).
- Delete a custom role for Scan. (page 656)



Built-in roles for Contrast Scan

You can select these built-in roles for user access groups:

> **NOTE**
> You cannot change the settings for the built-in roles.

- **RBAC Admin:** Includes all Scan actions for the organization. If you have an Organization Admin role, Contrast assigns this role to you automatically.
- **View all projects:** Includes these actions:
  - View all details for scans and scan projects.
  - Download scan results in a SARIF or CSV file.
- **Upload artifacts:** Includes these actions:
  - Upload files for scanning.
  - Start scans.
- **Manage all projects**: Includes these actions:
  - View all details for scans and scan projects.
  - Download scan results in a SARIF or CSV file.
  - Create scan projects.
  - Update scan project details.

Add roles for Contrast Scan

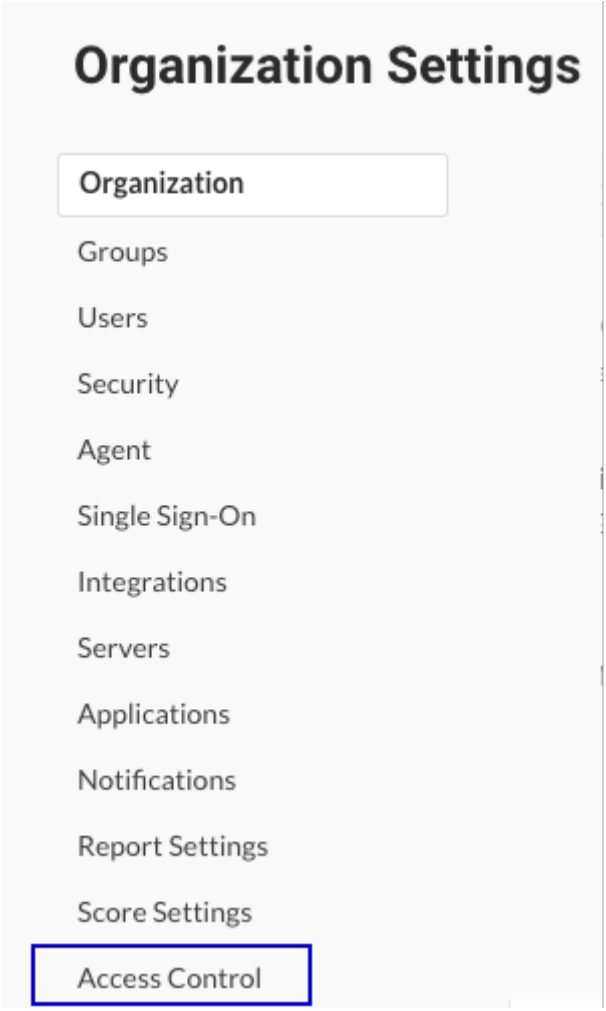
Create roles to customize which Scan resources and actions users can access.

Before you begin

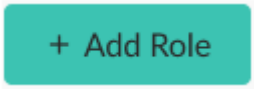
- An Organization Admin role is required.
- Determine if you need to add a custom resource group (page 649) for the role.

Steps

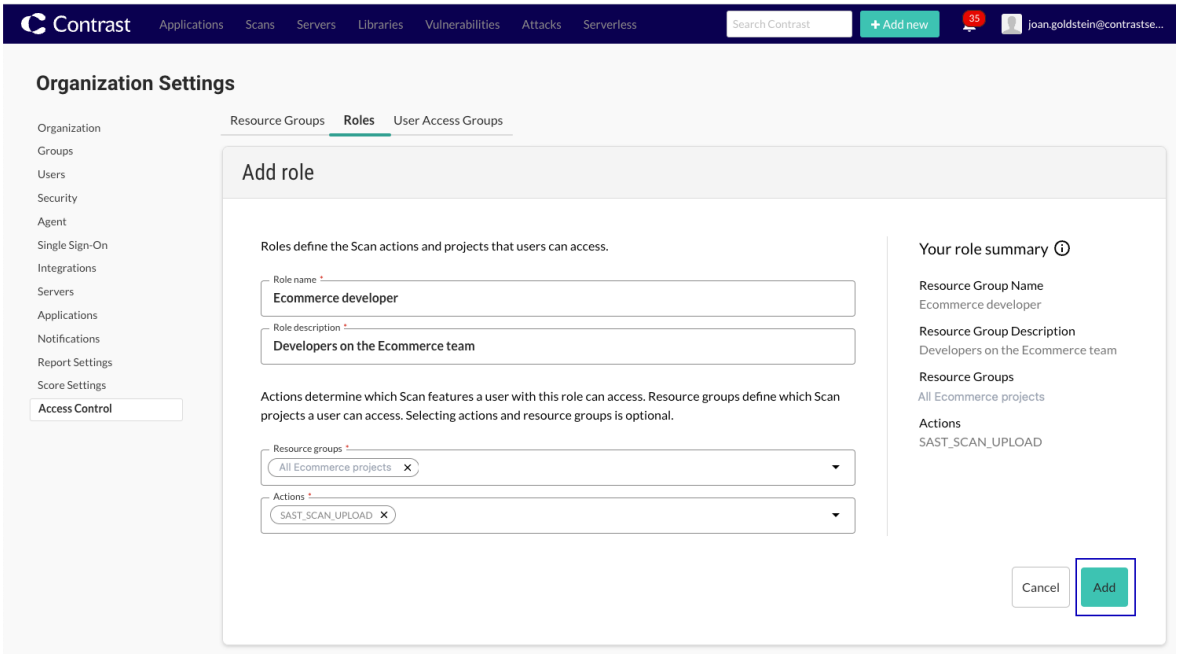
1. From the user menu, select **Organization settings**.
2. Select **Access control.**



3. Select the **Roles** tab.
4. Select **Add role**.



5. Specify the role settings:



a. **Role name:** Specify the name for the role.
   The name must be unique for the organization. Use up to 50 characters, including spaces and special characters.
b. **Role description:** Specify a description of the role.
   Consider using a description that indicates the purpose of the role. Use up to 1024 characters, including spaces and special characters.
c. **Resource groups:** Select one or more resource groups.
   Select a built-in resource group (page 648) or a custom group (page 649).
d. **Actions:** Select the Scan actions for the group:
   - **PLATFORM_ORG_MANAGE:** Manage all access control entities such as user access group, roles, and resource groups for organizations.
   - **SAST_PROJECT_MANAGE:** View scan details, upload artifacts, create scan projects, and update scan project details.
   - **SAST_PROJECT_VIEW:** View details for scan projects and scan details, including vulnerabilities, other findings, and policies
   - **SAST_SCAN_UPLOAD:** Upload files for scanning and start scans.

6. Select **Add**.
   You can now assign this role to user access groups.

Edit roles for Contrast Scan

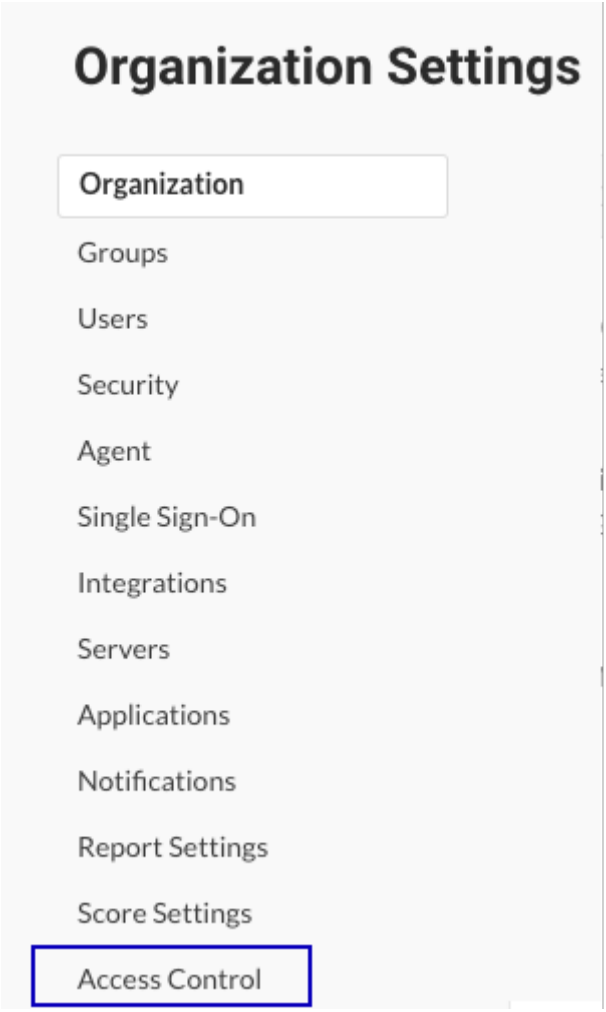
Use this procedure to edit or copy an existing custom role.

Before you begin

- An Organization Admin role is required.
- You can copy an existing role and edit it to add a new role.

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**.

Delete roles for Contrast Scan

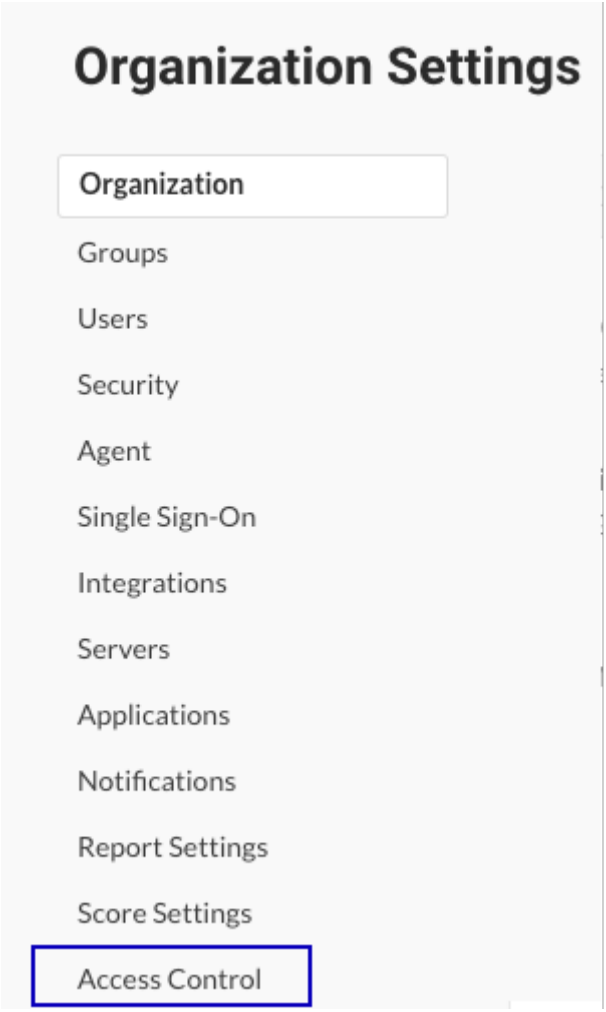
Use this procedure to delete a custom role.

Before you begin

- An Organization Admin role is required.
- You can delete only custom roles, not built-in roles.

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.
5. In the Delete role window, select **Delete**.

User access groups for Contrast Scan

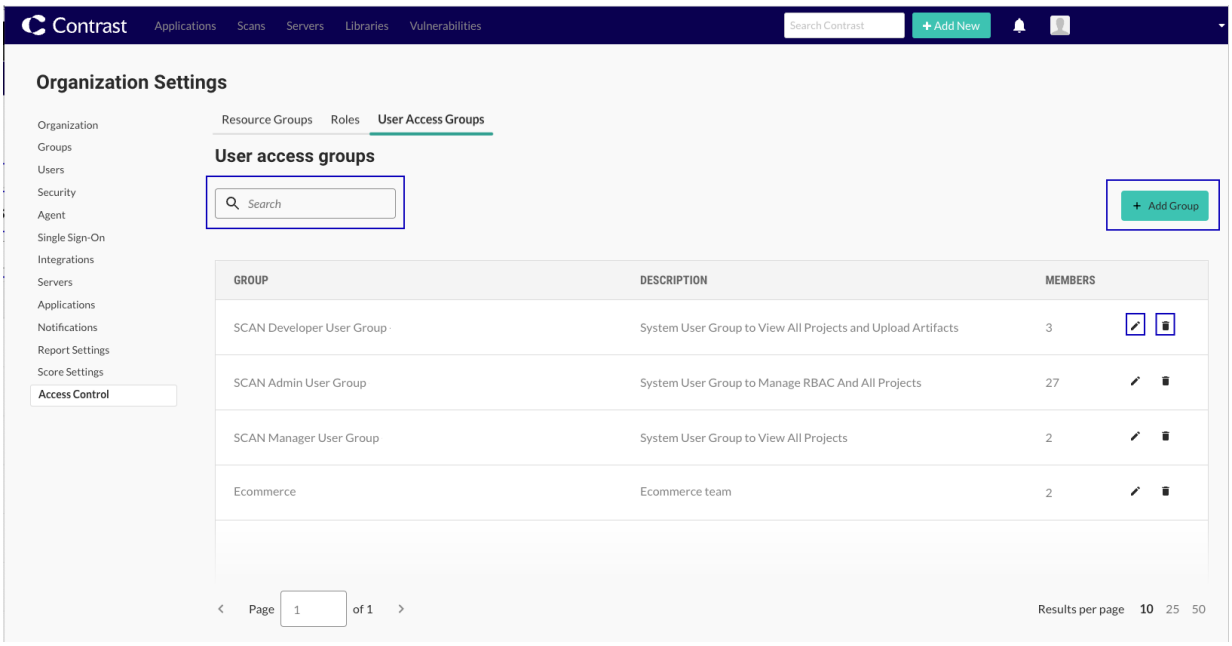
User access groups let you specify specific roles and resource groups for one or more users.

You can add users to built-in user access groups or create custom groups (page 658).

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 for Scan. (page 658)
- Edit a user access group for Scan. (page 660)
- Delete a user access group for Scan. (page 661)



Built-in user access groups for Contrast Scan

You can add users to these built-in user access groups:

- **Scan Admin Group:** Includes these roles:
  - RBAC Admin
  - Manage all projects in all organizations.
- **Scan Manager Group**: Includes this role:
  - View all projects
- **Scan Developer Group:** Includes these roles:
  - View all projects
  - Upload artifacts

NOTE

Other than adding users, you cannot change the settings for the built-in user access groups.

Add user access groups for Contrast Scan

Create user access groups to let a collection of users access Contrast Scan. The roles you assign to users in the group determine which Scan capabilities they can access.

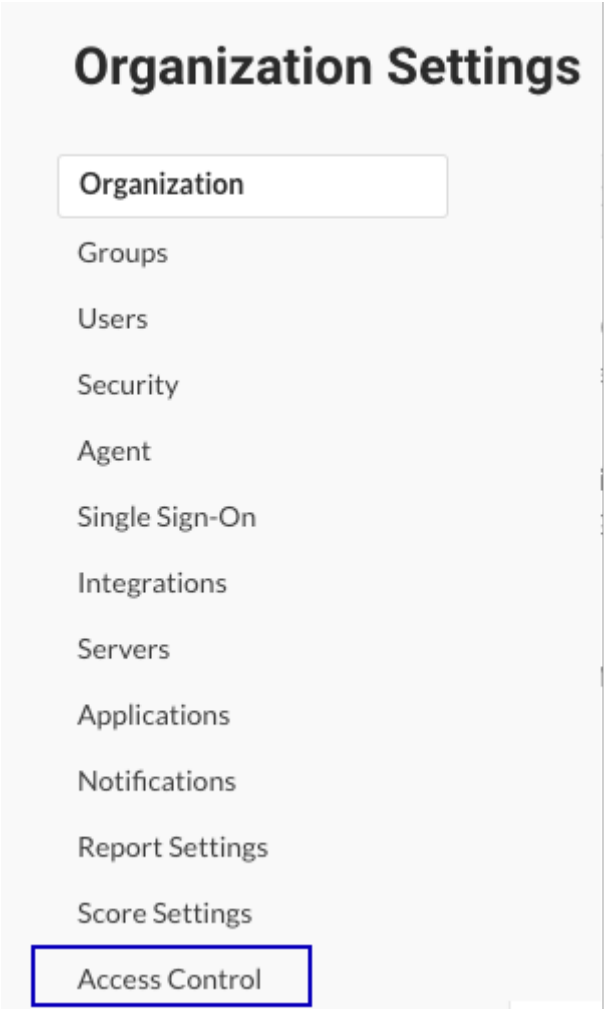
Before you begin

- An Organization Admin role is required.
- Add organization users (page 635), if necessary.
- Determine whether you need to add custom roles (page 653) or custom resource groups (page 649).

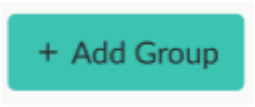
Steps

1. From the user menu, select **Organization settings**.

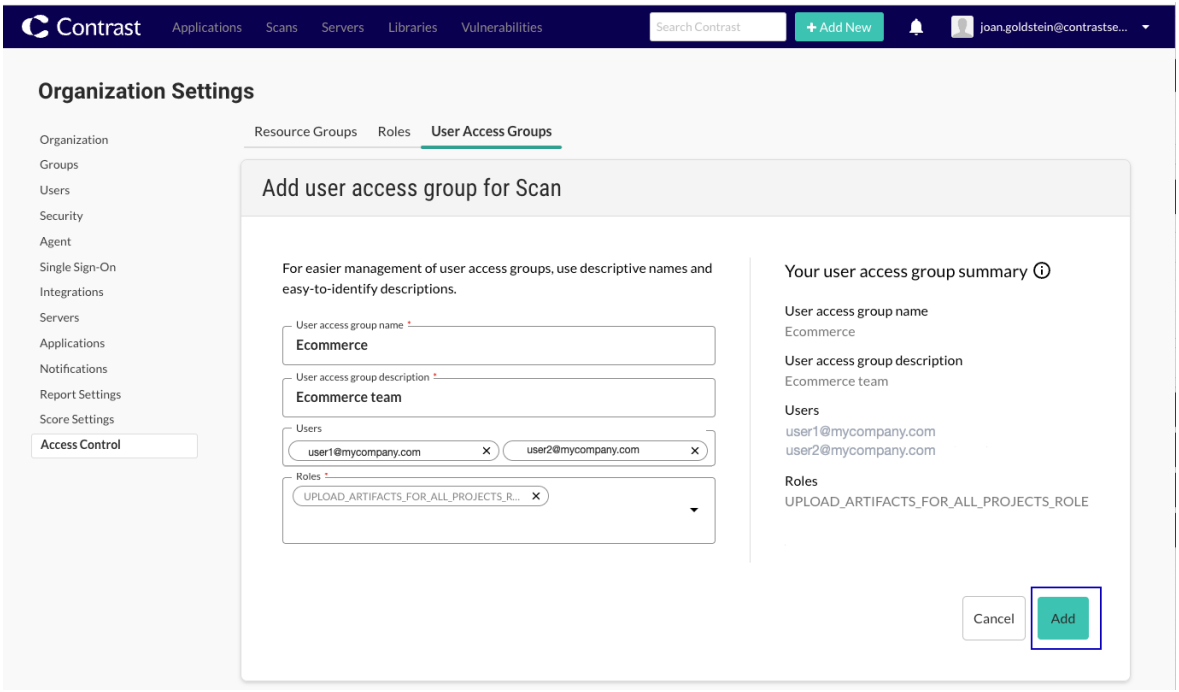
2. Select **Access control.**



3. In the User access groups tab, select **Add group**.



4. In the Add user access group for Scan window, specify these settings:



- **User access group name:** Specify a name for the user group.
  The name must be unique for the organization. Use up to 50 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 652)or custom roles (page 653).

5. Select **Add**.

Edit user access groups for Contrast Scan

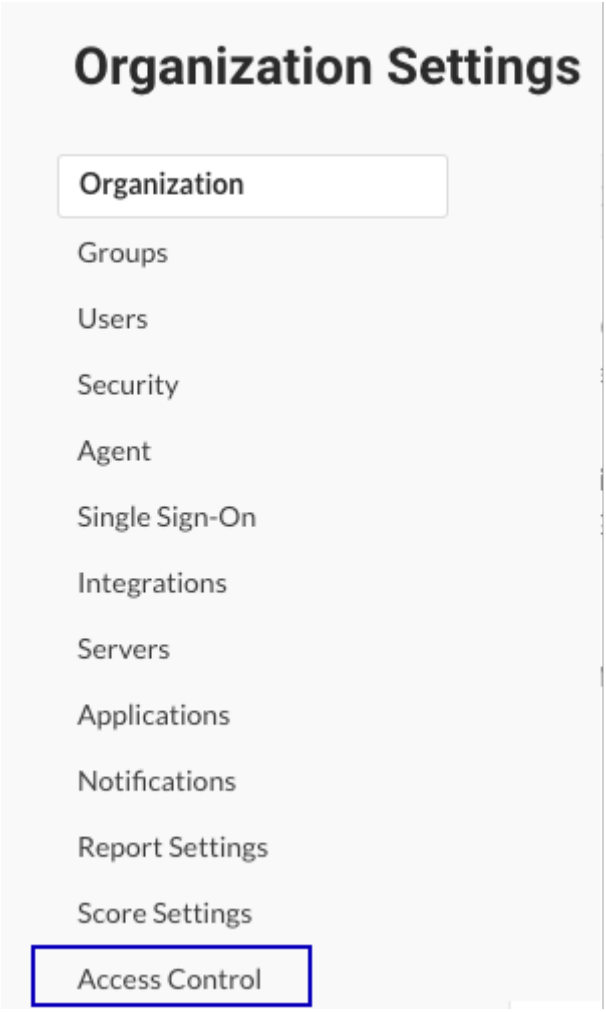
Use this procedure to edit an existing user access group. You can edit some or all of the settings.

Before you begin

- An Organization Admin role is required.

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 user access groups for Contrast Scan

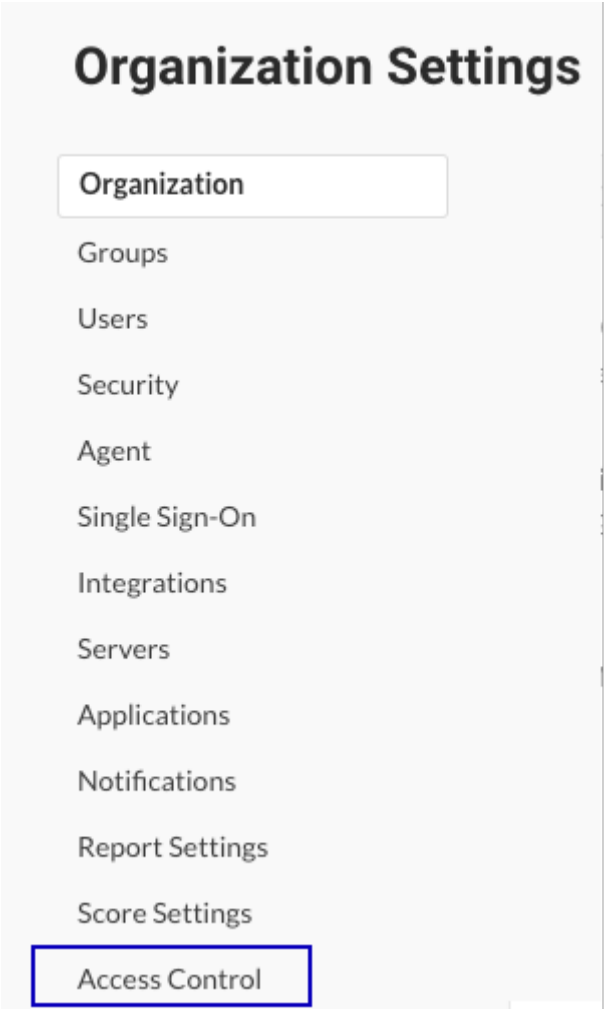
Use this procedure to delete a user access group.

Before you begin

- An Organization Admin role is required.

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**.

Require vulnerability approval

As an Organization Administrator, you can require administrative approval when closing vulnerabilities (page 605) 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 538).

To configure this requirement:

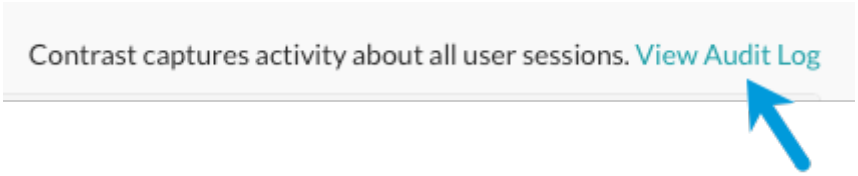
1. In the **user menu**, select **Policy management > Vulnerability management > Vulnerability behavior**.
2. Select the box next to **Require administrator approval when closing vulnerabilities**.
3. Choose the statuses and severities of vulnerabilities that should automatically go into a **Pending** 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.

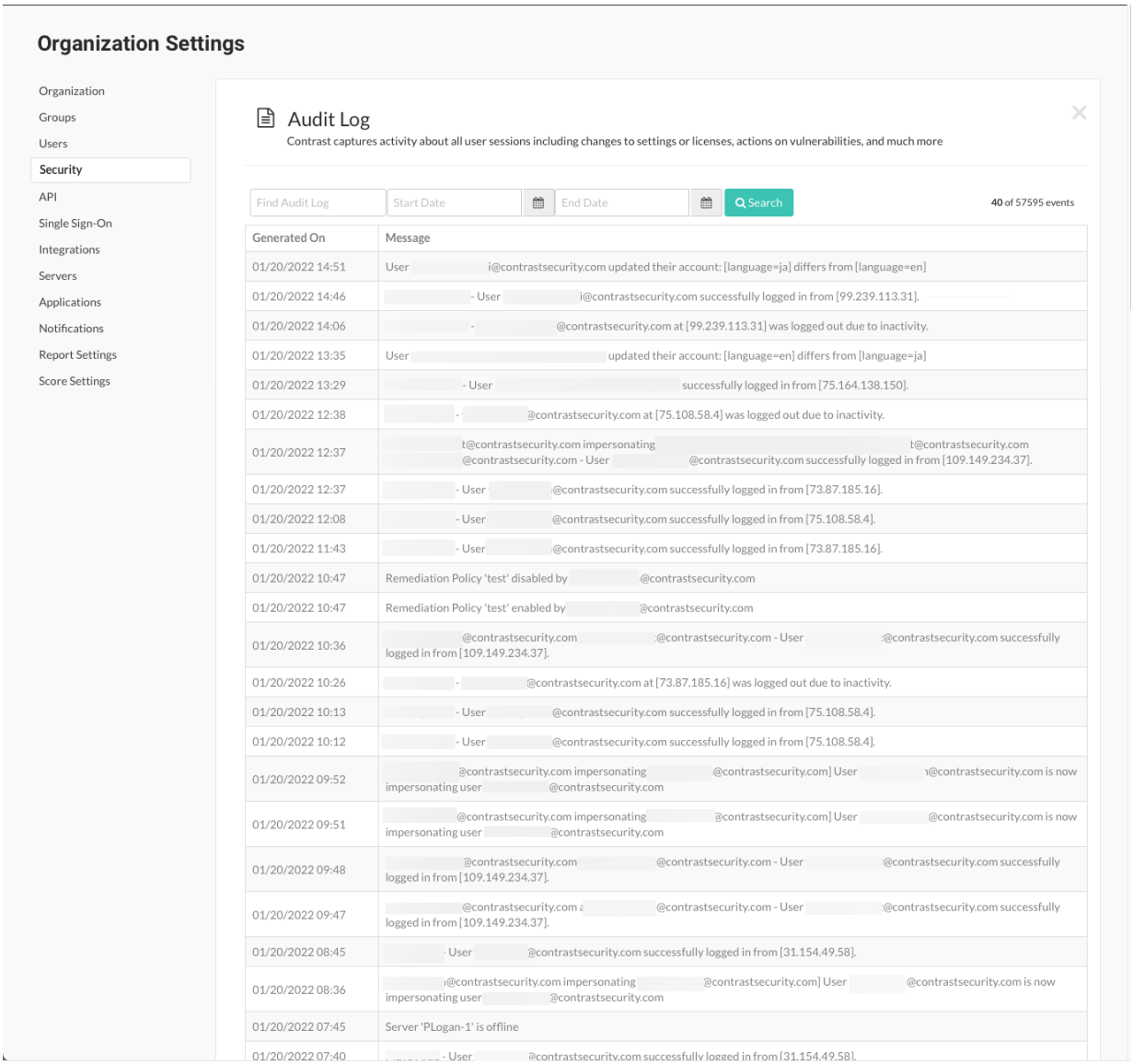
View audit log

Contrast captures activity about all user sessions including changes to settings or licenses, actions on vulnerabilities, and much more.

1. Under the **User Menu**, select **Organization Settings** > **Security**.
2. Click the **View Audit Log** option on the upper-right of the screen.



This opens the audit log page.



This is where you'll find information about:

Agents	Downloading the agent/agent launcher/contrast service
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	Creating reports Vulnerability trend reports
Servers	Changing defaults Creating notifications Deleting/Disabling/Updating/Removing licenses Enabling/Disabling Protect
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

TIP
Use the search fields to look for a specific log by date or name.

Impersonation

Impersonation lets someone with a system role (page 746) 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 700).
  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 666)

When you turn on impersonation:

- If you have a SuperAdmin role, you can use impersonation (page 710) 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 710) 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

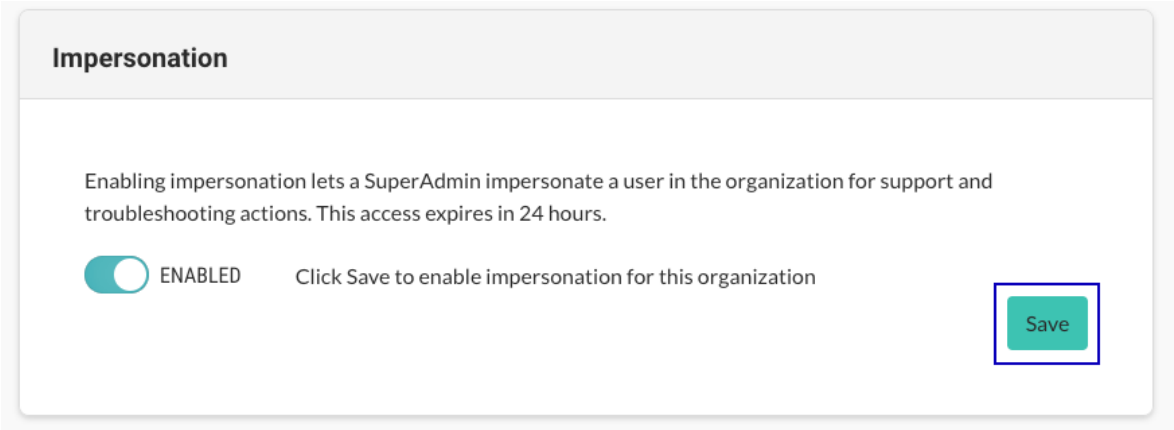
Use this procedure to turn on impersonation (page 666) for an organization.

Before you begin

- An Organization Admin role is required.
- 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 700) 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**.



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 668) or decide how to manage your system administration (page 699).

Get started on-premises

As an on-premises customer, you can set up your own instance of Contrast without 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 30).

For on-premises customers, to use Assess, Protect or both, you must complete at least two installations:

- Install Contrast (page 673)
- Install an agent for each application server (page 30)

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 669)
- Sizing recommendations (page 670)

After installation, you can further configure:

- Tomcat (page 685)
- JRE (page 686)
- HTTPS (page 686)
- Contrast settings (page 727)

For the long term, make sure you have a plan to:

- Manage system administration (page 699)
- Configure settings (page 727)
- Maintain the Contrast system (page 734)

Contrast installation

Options for installing and deploying Contrast in an on-premises environments include:

- Install with the Contrast installer (page 673)
- Use a distributed MySQL database (page 677)
- Deploy with a WAR file. (page 676)
- Deploy in a distributed environment (page 679)

Next steps

- Review the Contrast system requirements. (page 669)
- Review the Contrast sizing recommendations (page 670).
- Download the Contrast installer from the Contrast Hub or with curl commands (page 671).

Contrast system requirements

The following table lists the system requirements for installing the Contrast application.

Before you install Contrast:

- Download the Contrast Installer (page 672) from the Contrast Hub or by using curl commands (page 671).
- Review the sizing recommendations. (page 670)

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	• RHEL 8.6 • RHEL/CentOS 7 • Microsoft Windows 2019 • Ubuntu 18.04 LTS or higher (up to 20.04 LTS)	• RHEL/CentOS 7 • Microsoft Windows 2012 R2 or higher • Ubuntu 16.04 LTS	Support for CentOS 6 ended on December 1, 2020.
Java	Java is bundled with the installer.		
MySQL	• For on-premises customers using Contrast versions 3.8.11 or prior, MySQL 5.7.23 is bundled with the installer. • Beginning with Contrast 3.9.0, MySQL 8 is bundled with the Linux on-premises installer. • Beginning with Contrast 3.9.3, MySQL 8 is bundled with the Linux and Windows on-premises installer. If you experience issues, contact Support.		

IMPORTANT

- For on-premises customers using MySQL 8 (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, 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	11	11
MySQL	• 8.0.30 (Contrast 3.9.1 or later) • 5.7.23 (Contrast 3.9.0 or earlier)	• 8.0.27 (Contrast 3.9.1 or later) • 5.7.23 (Contrast 3.9.0 or earlier)

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 679). 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 679). 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 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 697).

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 673).

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 671) instead of the Contrast Hub to download installers and licenses.

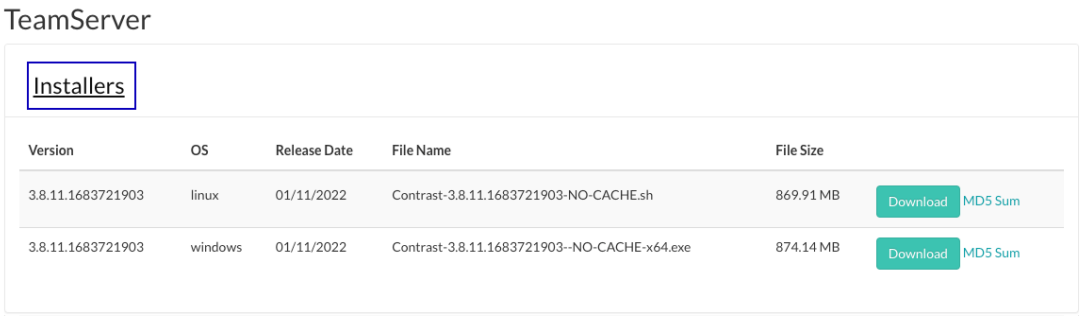
NOTE

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 697).

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:



   - **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 696).
   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 697) after installation.
4. Download the license file.
   The license file is configured with a SuperAdmin (page 746) 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 673)

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 30) without installing Contrast. Get started by installing an agent (page 30).

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 672) from the Contrast Hub or by using curl commands. (page 671)
- If you don't have internet access (air-gapped installations), download the library data manually (page 696).
- Review the system requirements (page 669) and sizing recommendations (page 670).

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 736) or configure the JRE (page 686)).
   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
`-h -help`	Shows help for common command line arguments.
`-c`	Forces the installation to run in Console mode.
`-q`	Executes the installer in Unattended mode.
`-g`	Forces the installation to run in GUI mode. (Windows only).
`-console`	If the installer is executed in Unattended mode and the `-console` argument is passed on Windows, a second console shows the output of the installer.
`-overwrite`	Forces the installer to overwrite all files in Unattended mode regardless of the overwrite policy specified in the installer. **CAUTION** This can cause your configuration to be overwritten back to default values.
`-dir`	Only valid in Unattended mode; specifies the directory where Contrast should be installed.
`-Dinstall4j.debug`	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 `stderr`.
`-Dinstall4j.keepLog=true` `-Dinstall4j.alternativeLogfile=[path]`	The installer creates a log file prefixed `i4j_log` for all installations and uninstallation in your temp directory. This log file can be helpful for debugging purposes. If your installer contains an **Install files** action and terminates successfully, the log file is copied to *[installation dir]/.install4j/installation.log*. Otherwise, the file is deleted after the installer or uninstaller terminates by default. When using the `-Dinstall4j.keepLog=true option`, the log file won't be deleted. With the `-Dinstall4j.alternativeLogfile=[path]` option, the log file is copied to the file specified with `[path]`. This should be an absolute path name. Neither option has any effect if the log file has already been copied to the installation directory.

Command line argument	Description
`-varfile (filename)`	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.
`--skip-preflight`	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 694), 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 683) to the user interface: the default SuperAdmin and the default Organization Adminstrator. 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 727) (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 697). If you don't have internet access (air-gapped installations), update library data manually (page 696).

Additional options

After you install Contrast, you have options for expanding your installation:

- Use a distributed MySQL database (page 677)
- Deploy with a WAR file (page 676)
- Deploy in a distributed environment (page 679)

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 673)

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 669)
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 679).

1. Install and configure a supported version (page 669) of MySQL on the database server host.
2. Create a maintenance window for Contrast downtime.
3. Back up the embedded MySQL database (page 736).
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. Create the Contrast database with `create database <jdbc.schema>;`.
6. Grant permissions to the Contrast user with `GRANT ALL PRIVILEGES ON *.* to 'contrast'@'%';`.
7. Exit from MySQL.
8. 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>
```

9. Update the configuration in the encrypted properties editor (page 734). 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.

10. 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= ""
    ```
11. Restart Contrast (page 683).

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 "*" above for illustration, 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.

Distributed deployment of Contrast

A distributed configuration of Contrast deploys the database and application server to separate servers. Use a distributed configuration if you:

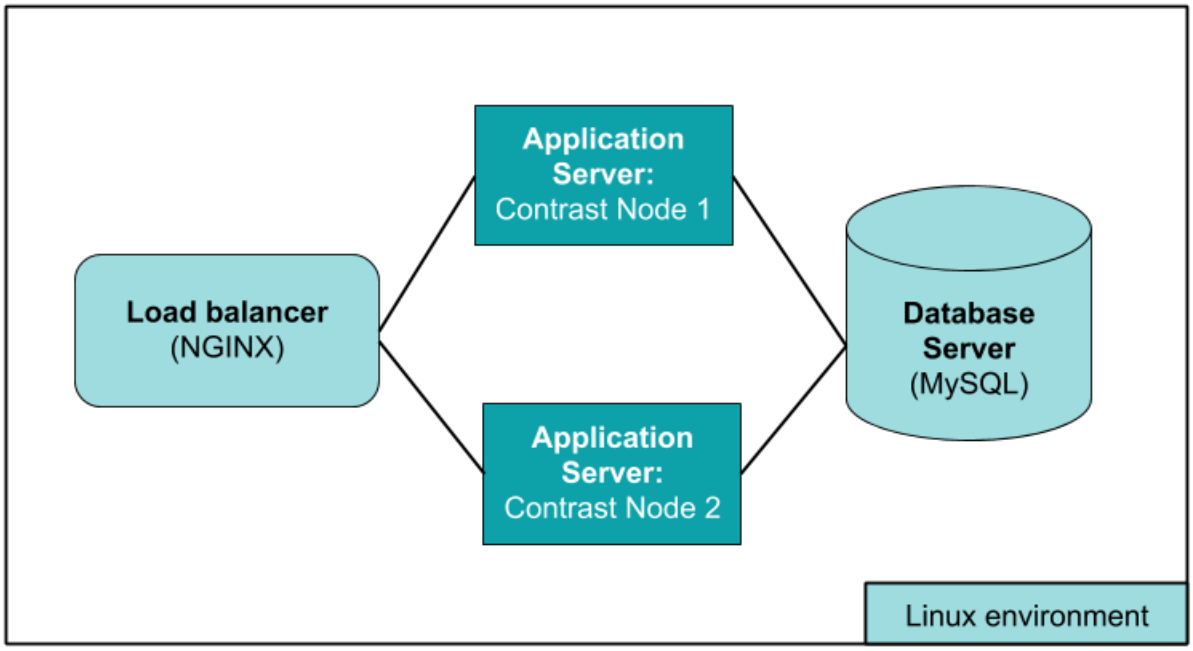
- Plan to use more than 100 connected agents
  Without a distributed configuration for this situation, you are likely to experience performance issues.
- Want to use load-balancing for better performance and scalability
- Require additional administration and management

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 677).
  - Get Contrast application version numbers by looking in the `VERSION` file in `/usr/local/contrast/VERSION` . Get the Contrast database version using this command:

    ```
    select `version` from schema_version ORDER BY `installed_on` DESC \
    LIMIT 1;
    ```

    You need the version numbers to determine the order of tasks.
    For example, if the application version is 3.3.2 and the database is version 3.3.2.012, you can say the versions are the same because it's safe to drop .012 from the database version number. Because of this, an existing application server A can run with a separate database B on version 3.3.2. If you want to install application version 3.4.2 onto a new application server C and connect it to database B, you will need to either stop or update server A before installing version 3.4.2 on server C.
- If you are **new to Contrast**, you should:
- Install and configure MySQL on the database server according to Contrast system requirements (page 669). Create a MySQL user who has permissions to connect remotely. You will use these credentials to connect to Contrast.
- Install Contrast (page 673) on **Application Server 1** with a distributed database configuration.
- Select the option for a distributed MySQL instance. Point the database creation to the database server, for example:

```
Choose a MySQL database configuration.
Default [1, Enter], Distributed [2]2
Host
[localhost]
```

```
<enter hostname of Mysql server>

Port
[13306]
3306

Credentials
Username
contrast

Password
<enter mysql password for contrast account>
```

Set up distributed servers

1. Create a TAR file from the Application Server 1's configuration files:
   - `data/conf/`
   - `data/esapi/`
   - `data/contrast`
   - `data/initialized`
   - `data/contrast/lic`
   - `webapp/Contrast.war`
   - `VERSION`

   Use a command similar to the following example:

   ```
   $ cd /usr/local/contrast
   $ tar -czvf ~/ctdc.tar.gz data/agents data/conf data/
   contrast.lic data/esapi/ data/.initialized data/.contrast webapp/
   Contrast.war/.contrast VERSION
   ```
2. Install Contrast (page 673) on **Application Server 2**.
   Complete the installation process as a privileged user.
   - On Windows, right-click on the installer and select **Run as administrator**.
   - On Linux, use a `sudo` command to launch the installer.

   IMPORTANT

   ```
   Full or Application Only Install
   Choose the desired installation type:
   Full install [1, Enter], Application Server Only [2]
   2
   ```

   Select **Application Server Only** and use the TAR file you created in the previous step. Do the same for any additional application servers you want to add to the distributed configuration.
3. Test the default users (page 683) created by the application to be sure they work with both Contrast **Application Servers** (**1** and **2**).
4. 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.

5. 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.

6. Deploy the WAR file:
   Create a symbolic link, copy, or move the Contrast WAR file to the Tomcat `webapps` directory.
   - This example shows the command that creates a symbolic link in a Ubuntu installation:

     ```
     $ sudo ln -s /opt/contrast-data/webapp/Contrast.war /var/lib/tomcat7/
     webapps/Contrast.war
     ```
   - This example shows the command to copy the WAR file to the Tomcat `webapps` director in an Ubuntu installation:

     ```
     $ cp /opt/contrast-data/webapp/Contrast.war /var/lib/tomcat7/webapps/
     Contrast.war
     ```
7. 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 673) 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 746).
  - **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 673) 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 435) or by using command line on Windows (page 727) or Linux (page 726).

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 736) 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 685)
- Configure the Java Runtime Environment (page 686)
- Configure HTTPS (page 686)
- Customize MySQL (page 688)
- Configure reporting storage (page 690)
- Configure logs (page 692)
- Use Redis as a shared cache (page 693)

Configure Tomcat

During installation (page 673), 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 673), you are given the option to use either an embedded JRE or a pre-installed JRE.

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\jre11
```

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.

Use the Contrast HTTPS connector

Use these steps to create a Java KeyStore (JKS), with a signed certificate, that your on-premises Contrast application server will use at runtime.

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
```

> **IMPORTANT**
> When you create the JKS and the private key, it is important that both passwords are the same. If the passwords are different, the HTTPS connector doesn't work.

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.

> **NOTE**
> In order for Contrast to use the SSL Certificate, the certificate can't be protected with a passphrase.

- **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 PCKS #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
```

NOTE

If you retrieve your certificates through another method, you may need to create a keystore differently. For example, if you end up with a:

- `server.crt`,`priv.key` and `inter.crt` files: Convert the files to a PKCS #12 and create a keystore using 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
```

- PKCS #12 file: Create a keystore with this command.

```
keytool -importkeystore -srckeystore cert.pfx -
srcstoretype pkcs12 -destkeystore contrast.jks -
deststoretype 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>`, `<file>`, `<password>`, `<hostname>` with your port, JKS file path, password, and the hostname alias given in the keytool command.

```
https.enabled=true
https.port=<port>
https.keystore.file=<file>
https.keystore.pass=<password>
https.keystore.alias=<hostname>
```

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.

NOTE
It is also possible to use the HTTPS connector with a self-signed certificate.

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.)

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 682) 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.vmoptions` 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 (prerferred method)**
  1. In the Contrast `bin` directory, `c:/Program Files/Contrast/bin` or the default directory `/opt/Contrast/bin` ,open `contrast-server.vmoptions`.
  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.

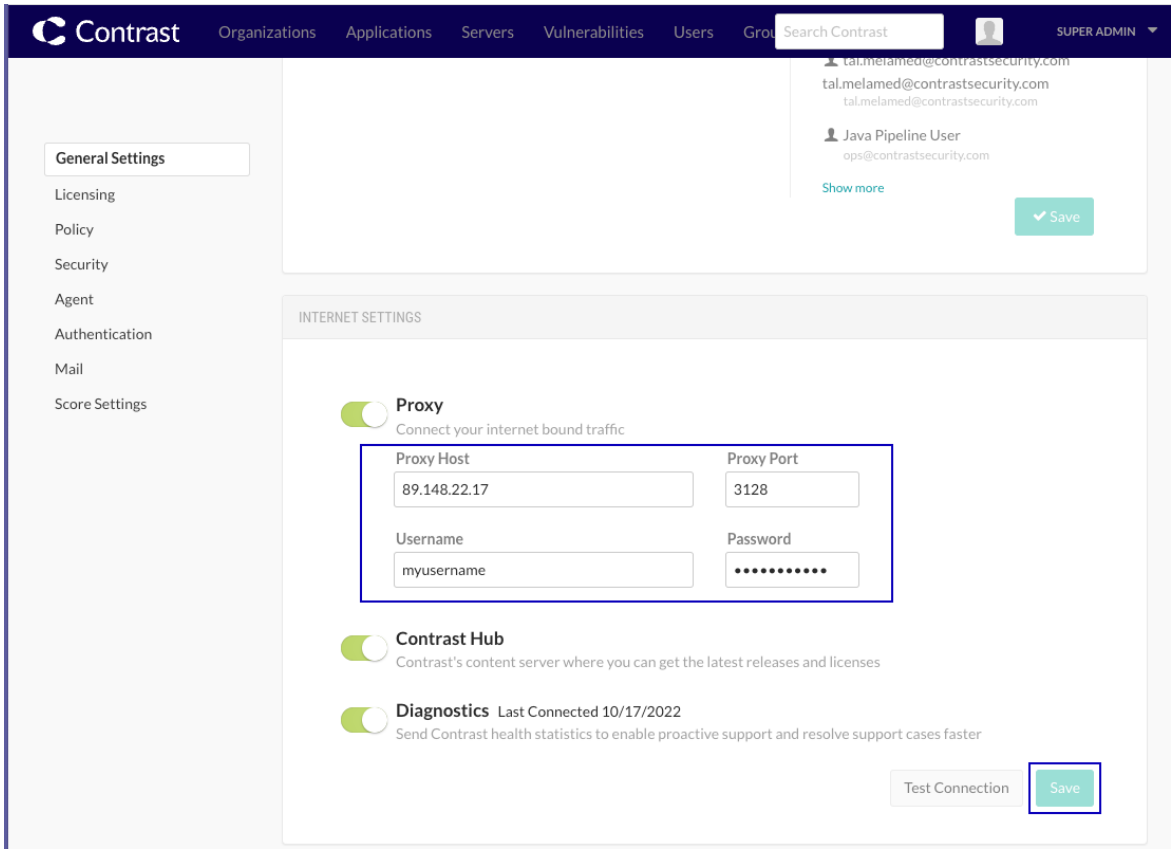
For example, if you have an on-premises Jira installation as well as an integration for MS Teams in the cloud, you can use the `http.nonProxyHosts` argument to exclude the Jira integration from using the proxy server, as shown in the following example:

```
-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 value options are `DB` and `FILE_SYS`.
- **`reporting.storage.path`:** This is required when storage mode is set to `FILE_SYS`.

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\\datareports
```

With the default configuration, 1,250 vulnerabilities can be exported from an attestation report. In some cases, a user may want to run a larger report containing more than 1,250 vulnerabilities. Depending on the size of the instance, Contrast may run into heap space issues.

To decrease or increase the limit, set the `reporting.generation.limit` property in the `general.properties` file, and then restart Contrast.

Contrast logs

For the Contrast application, Log4j version 2 is used as the logging framework.

You can configure logging thresholds (page 692),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
audit.log	Logs audit events such as: • Logging in and out of the application • Impersonating another user • Switching organizations • Accessing the administrator portal • Changes to the configuration and settings of Contrast by a SuperAdmin account • User account service issues (locked accounts, password changed, etc.) • Deleting traces • Changes to a license or an expired license notification • API Key changes
console.log	Default application event log
contrast-error.log	Logs messages printed to `stderr`
contrast-stdout.log	Logs messages printed to `stdout`
contrast.log	Like Tomcat's stdout or console log, contrast.log shows most key events happening inside Contrast to inform or help with debugging. It includes information about: • applications • servers • libraries • traces • users • Java stack traces for debugging purposes when a server exception takes place
security-events.log	Formerly the *esapi.log*, this log file is used for capturing key events from Contrast, such as the loading of a given property file.
migration.log	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 691).

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.useredis`	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.useredis=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 683).
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 694)
- Upgrade agents (page 695)
- Update IP address (page 696)
- Update library data (page 696)
- Upgrade library data automatically (page 697)
- Update license (page 698)

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 722).

Steps

1. Create a MySQL backup (page 736) 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 *$CONTRAST_HOME/data/conf*.
2. Be sure the Contrast application is running when you start the installation process. Agents will continue to send vulnerability and library messages during this time. When the application initiates a shutdown on its own, the agents defer sending messages until it can reach the application.
3. The upgrade process is nearly identical to the original process to install the Contrast application (page 673); however, you will be asked to choose to update an existing installation or perform a new installation. You should choose to update an existing installation.
4. The upgrade will initially perform a database backup. Depending on the size of your database, this process can take a few seconds to a few minutes. During this operation, the application should be accessible to agents and end users.
5. The update then deploys a new file system under the installation directory. This primarily consists of deploying the Contrast.war file directory to the *$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:** Download from Debian (page 414) or RPM (page 414).
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 683) 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

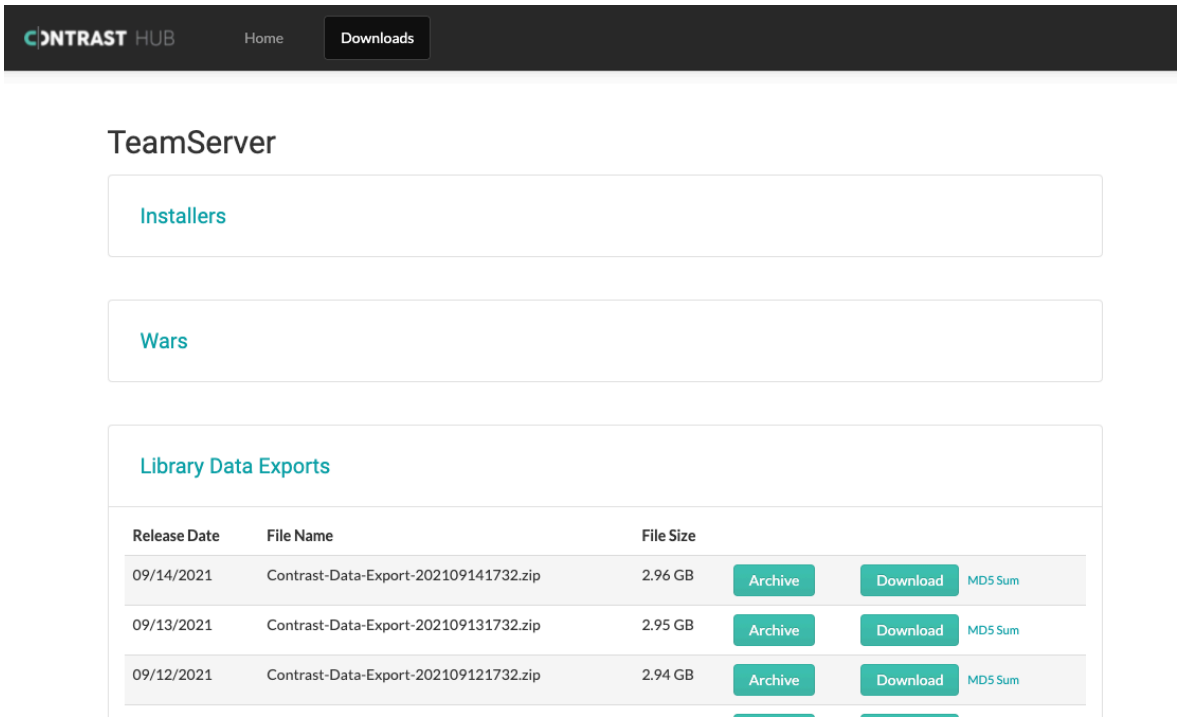
- A Contrast Hub account is required.
- For optimal performance, plan to download the library data on a monthly basis.
- If you have a distributed deployment (page 679) with multiple servers, use the procedure in this topic for one instance. You can use the same downloaded library data files on multiple 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.



4. Extract the downloaded ZIP file and place the CSV files in the *data/libraries* directory.
   Some files may be hidden due to their names, so ensure all extracted files are moved to this directory.
5. Restart Contrast. (page 683)
   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.

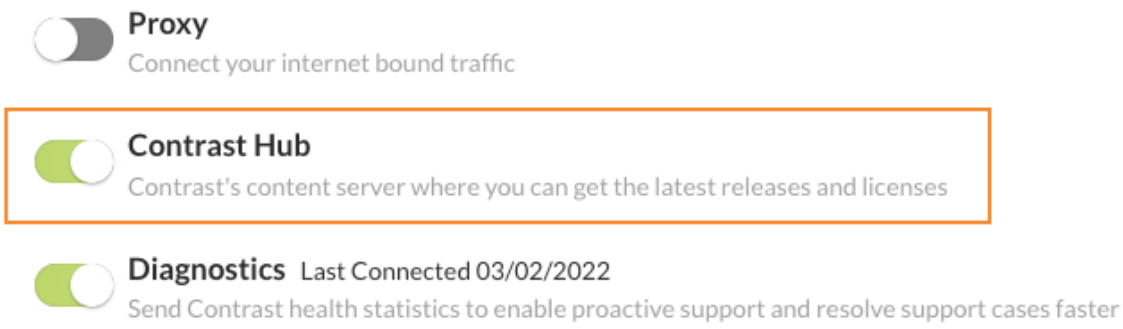
Contrast updates the library data approximately every 24 hours; CVEs are updated every 30 minutes. Your Contrast installation pulls the data from a Contrast database hosted on the cloud.

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.**



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 683) 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 683) 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 *myslqld* 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 746) 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 705).

- 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 705) 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 636)) 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

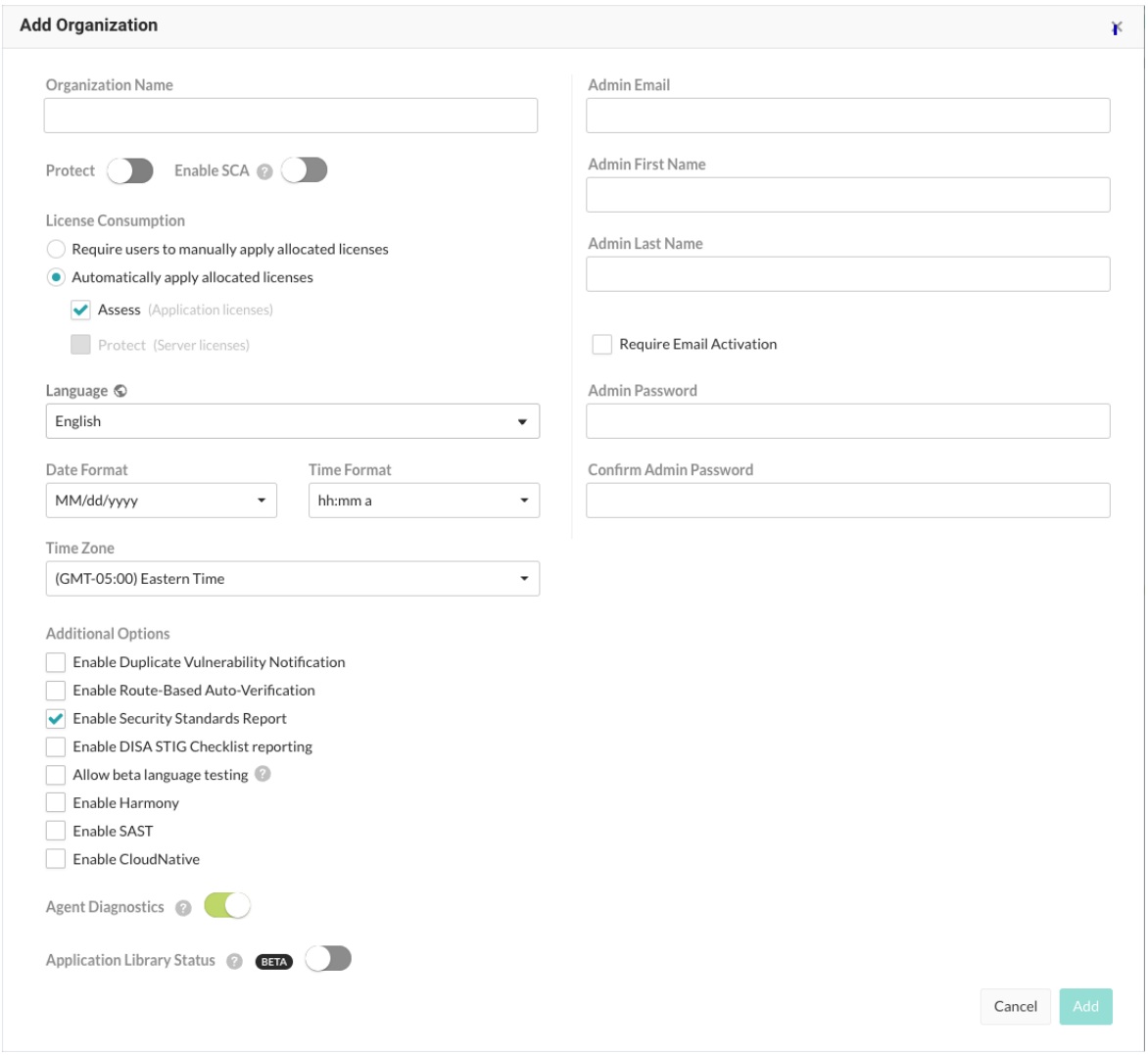
In Contrast, an organization is a group of associated users and applications with a shared business purpose. Contrast uses multi-tenant architecture: each Contrast customer is a tenant, represented as an organization.

Before you begin

- To create an organization, a System Administrator (page 746) role is required.
- All organizations require a unique name as well as an Organization Administrator (page 744) 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:



- 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 606), DISA STIG checklist reporting (page 549), diagnostics (page 729) 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.

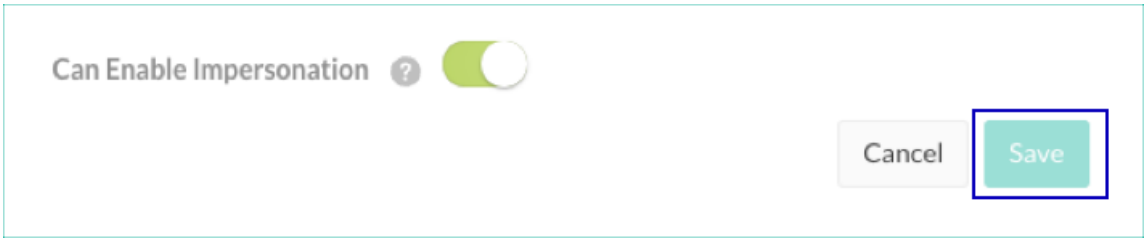
4. Select **Add** to create the organization. You may continue to create as many organizations as you need for multi-tenant support.

- **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 702), or bulk add multiple users (page 703). 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 727).
- **Groups and permissions:** Access and permissions are determined by roles (page 742). Most roles are assigned with access groups (page 705). SuperAdmin and ServerAdmin roles are designated differently (page 705). You can also grant Protect permissions (page 707) at a system level or when adding a new user (page 702) or bulk adding users (page 703).

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 713) or LDAP (page 717) 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 705) 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 746)** 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 707) 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

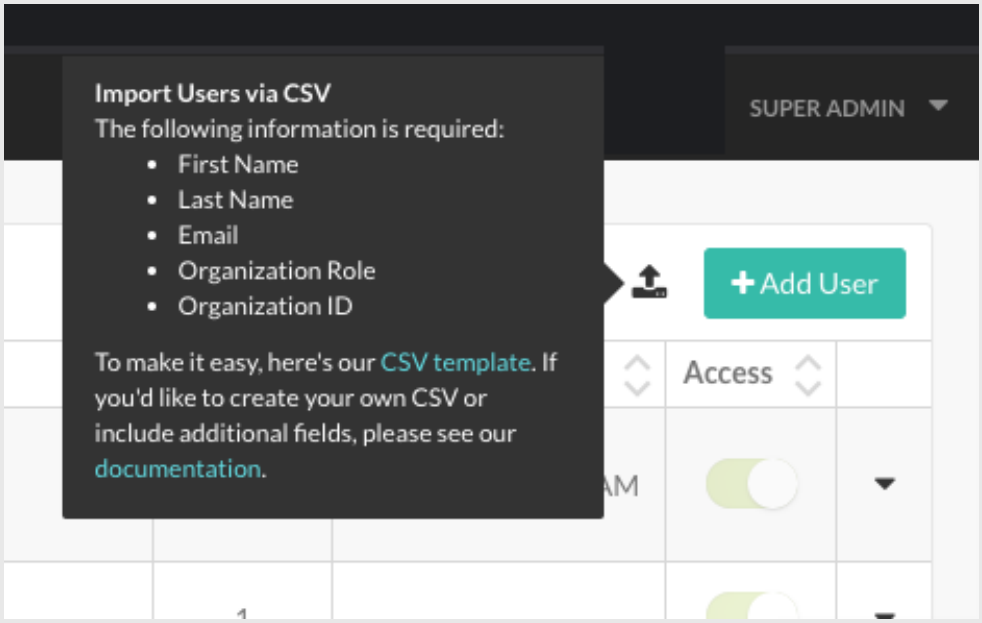
1. Go to the Users list:.
   a. For hosted customers, under the user menu, select **Organization settings** and then, select **Users**.

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
First Name	Required	User first name
Last Name	Required	User last name
Email or **Username**	Required	If you are using the Contrast's default internal directory, enter the user's email. If you are using an external directory, change **Email** to **Username** in the CSV, and enter usernames that match those in your external directory exactly.
Organization UUID	Required for on-premises customers	Get this value from the organization's general information settings (page 633).
Organization Role	Required	Values can be `View`, `Edit`, `Rules_admin` or `Admin`.
Date Format	Optional	The default value is the organization setting, such as `MM/dd/YYYY`.
Time Format	Optional	The default value is the organization setting, such as `hh:mm a`.
Timezone	Optional	The default value is the organization time zone.
Protect	Optional	The default value is `Off`.
Groups	Optional	Values can be `View`, `Edit`, `Rules Admin`, `Admin` or custom group names. Format multiple group names as `GroupA&&GroupB&&GroupC`.
Language	Optional	The default is the value configured for the organization (page 633).
System Administration	Optional	The default value is `Off`.
Email Activation	Optional	If the value is `None`, the default is `Required Password`.
Password	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 636). 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 746) 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 437), 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 746), organization (page 744) and application (page 743) 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)
Application Editor for all applications	**Application Viewer** for all applications
Organization Viewer for all applications	**Application Administrator** for the Red application
RulesAdmin for the Red application	**No Access** 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.

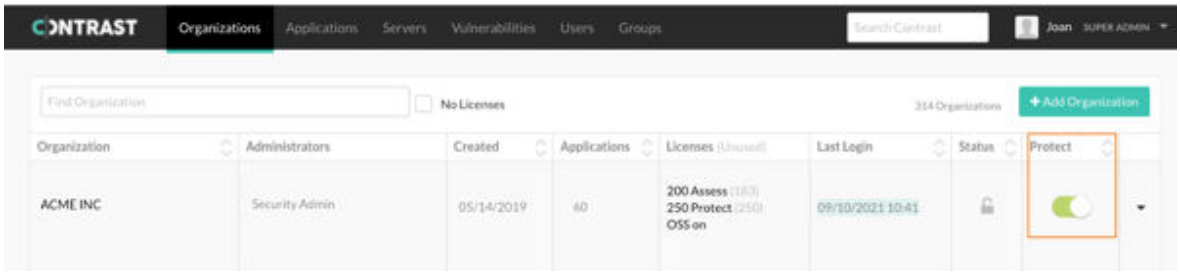
Before you begin

- 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.

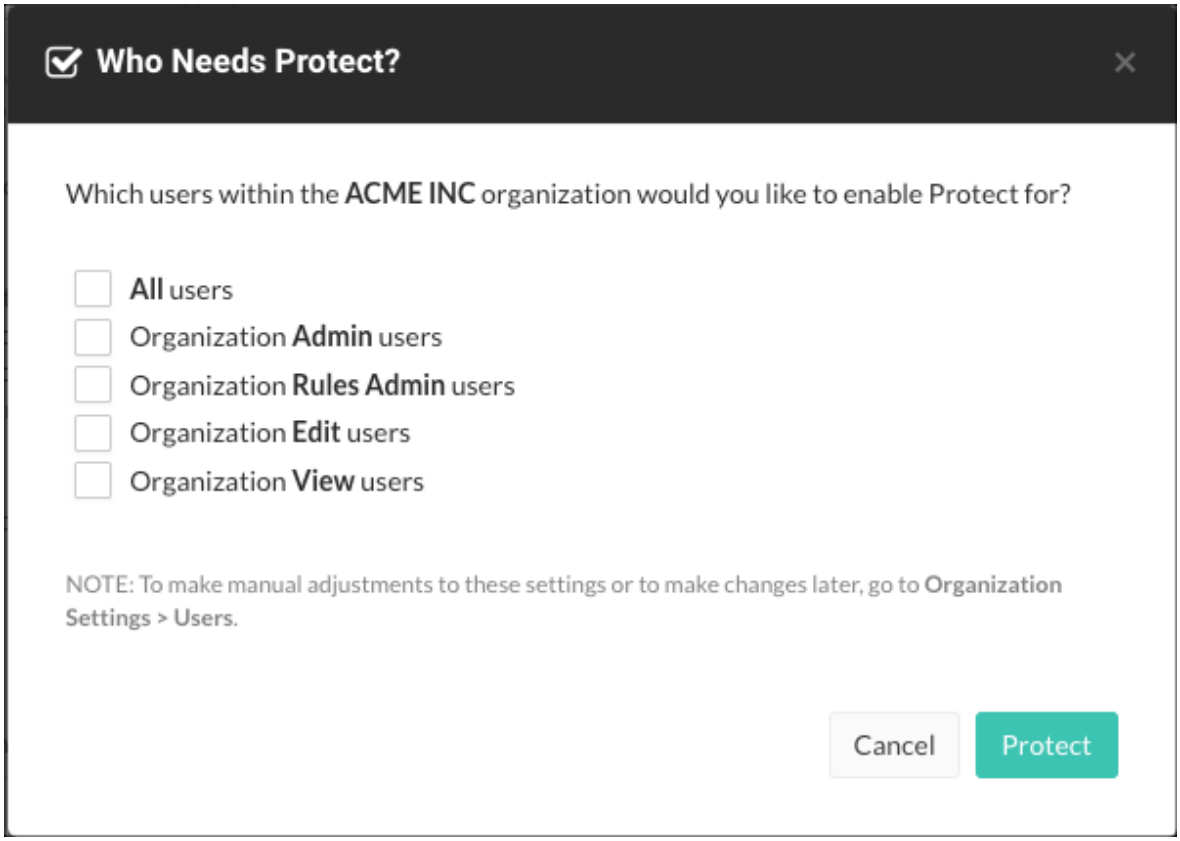
Steps

1. Log in as SuperAdmin.
2. Select **SuperAdmin** in the user menu.
3. Select **Organizations** in the header.

4. Find the organization for which you want to enable Protect. In the Protect column for that row, turn the setting on.



5. 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 702).
6. 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).

1. 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.

2. Then in Contrast, under organization settings (page 632), 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

```
1<saml2:Attribute Name="http://schemas.xmlsoap.org/ws/2005/05/identity/
claims/givenname"
2                             NameFormat="urn:oasis:names:tc:SAML:2.0:attr
name-format:unspecified"
3                             >
4               <saml2:AttributeValue xmlns:xs="http://www.w3.org/2001/
XMLSchema"
5                                     xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance"
6                                     xsi:type="xs:string"
7                                     >Dan</saml2:AttributeValue>
8           </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 673) 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 746).
  - **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 673) 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 435) or by using command line on Windows (page 727) or Linux (page 726).

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.

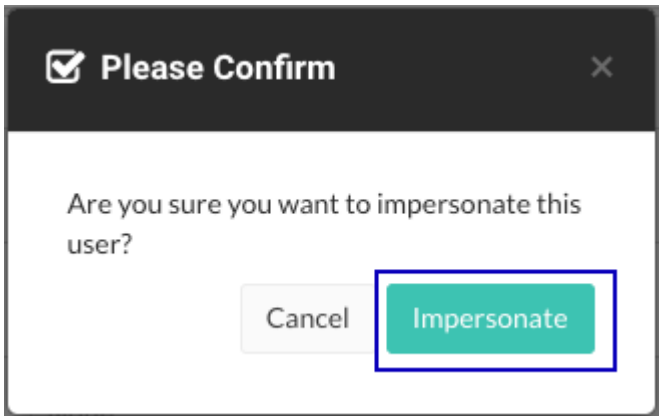
Before you begin

- The user you choose to impersonate must have an organization role (page 744).

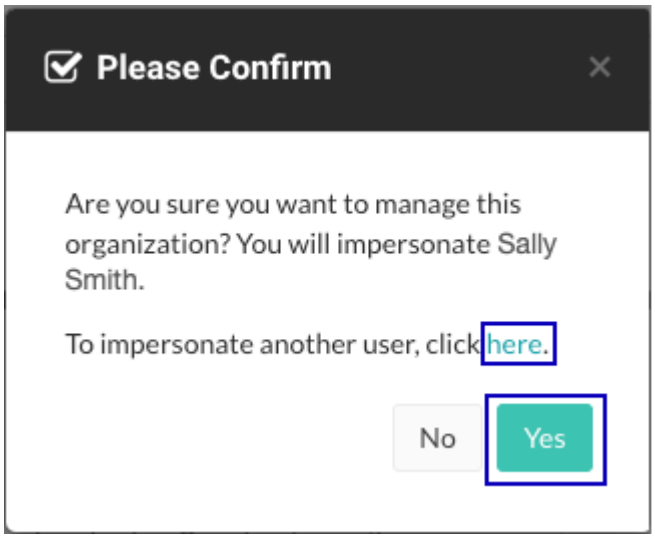
- If the Impersonation option is not visible for an organization, a SuperAdmin must turn on the Can enable impersonation (page 700) setting for the organization.
- An Organization Admin must enable impersonation (page 666) for the organization that you want to access.

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 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 725) and two-step authentication (page 713) 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 717)
- Active Directory (page 713)
- Single Sign-On (page 722)
- Trusted HTTPS Proxy (page 725)

Any changes to the authentication settings require that you restart Contrast (page 683). To change authentication, select **Authentication** under system settings (page 727).

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.

Enable two-step authentication at a system level

To enable or disable two-step authentication:

1. Under system settings (page 727), select **Security** in the left navigation.
2. Turn the toggle on (green) to enable two-step authentication.
3. Select the box next to **Allow organization override** to allow Organization Administrators to choose whether or not to require the feature for users (page 639).

NOTE
If a user belongs to multiple organizations, their default organization determines their two-step authentication settings.

A user can also choose how they want to receive two-step authentication notices. (page 436)

Configure Microsoft Active Directory

As a System Administrator (page 746), 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.

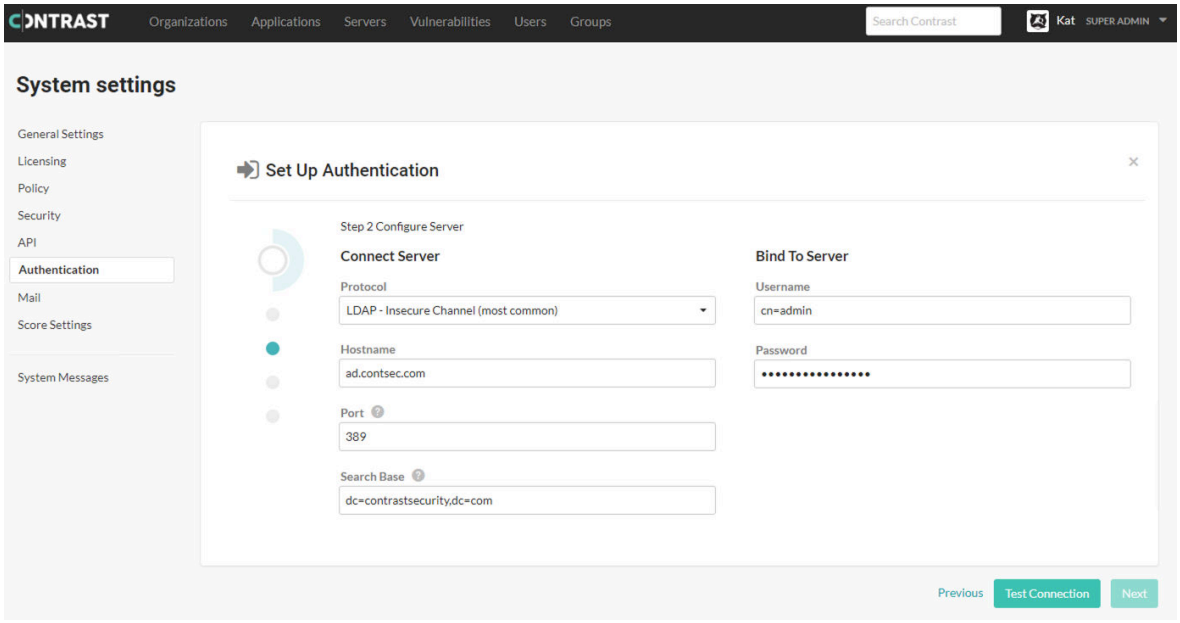
Access if AD is offline

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 716) 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 715) in Contrast.
3. Under system settings (page 727), 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 717).
- **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 715)
9. Configure advanced settings. (page 716)
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 716). Once you switch the setting, you should be able to verify login functionality more quickly.

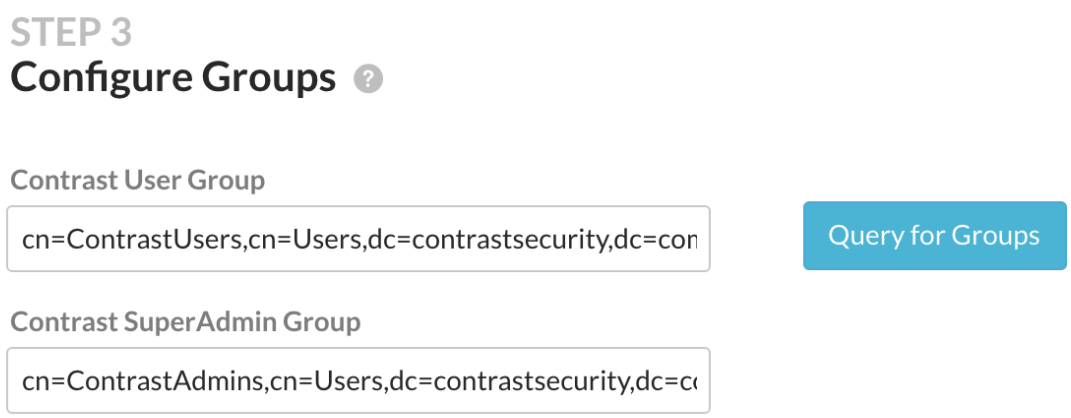
11. Select **Finish** to complete the configuration.

Configure groups for Active Directory

As part of Active Directory configuration (page 713), 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



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 635) 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.

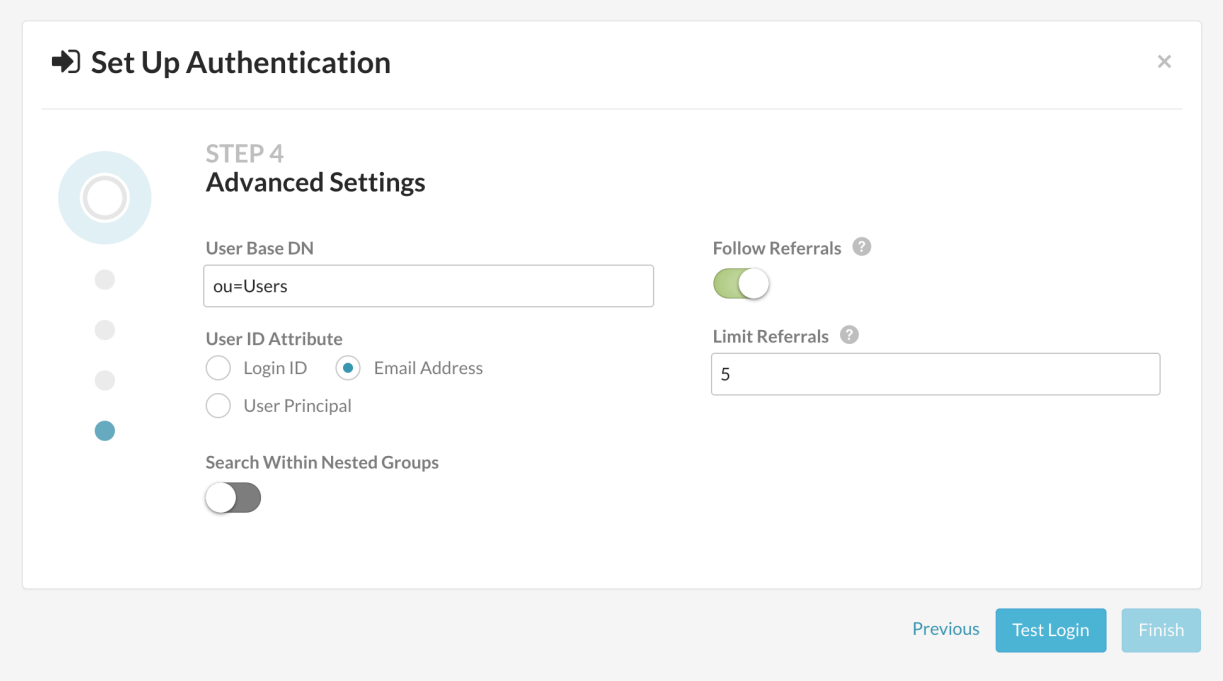
NOTE

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 713), under **Advanced settings** enter:



- **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 `userPrincipal` 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 713) 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:

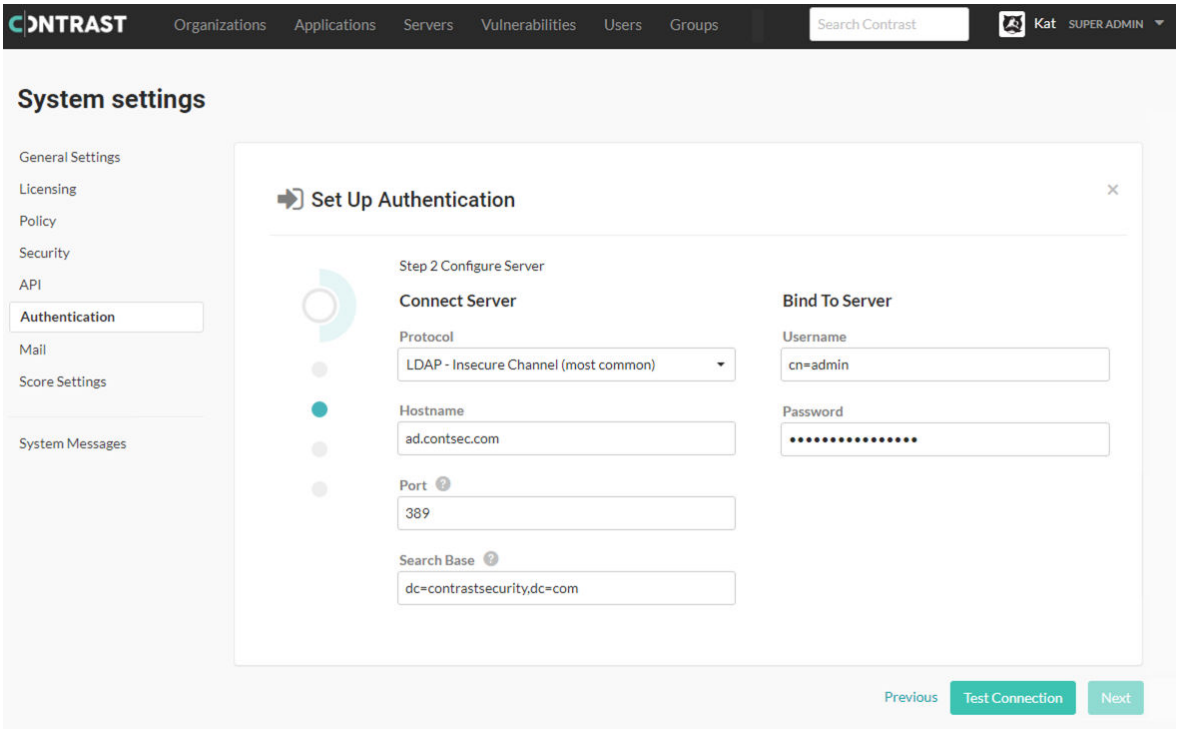
IMPORTANT
If you configure your LDAP integration to connect to your server using SSL, you may need to take extra steps for self-signed or privately signed certificates (page 722).

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 721) 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 719) in Contrast.
3. Under system settings (page 727), select **Authentication**.
4. Select **Change authentication method**.
5. Select **LDAP**.
6. Enter required information under **Connect server** and **Bind server**.



Option	Description	Default
Connect server		
Protocol	The protocol that should be used to communicate with the LDAP server. Choose between **LDAP** or **LDAP with SSL (LDAPS)**.	**LDAP**
Hostname	Enter the hostname to use when communicating with the LDAP server.	`localhost`
Port	Enter the port to use when communicating with the LDAP server.	**389 (LDAP), 636 (LDAPS)**
Search base	Enter the base distinguished name (DN) to use when communicating with the LDAP server. If your login domain is *yourdoman.com*, your base DN would be `dc=yourdomain,dc=com`.	`dc=contrastsecurity,dc=com`
Bind to server		
Method	Select the method to use when connecting to the LDAP server. Options are shown in the next table.	Simple
Username	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 **Username** 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 challege, which is sent with the MD5 hashed username and password to be authenticated.	Username, Password	SASL Realm

7. 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.
8. Configure groups for LDAP (page 719).
9. Configure users for LDAP (page 721).
10. To verify that the group and user mappings are correctly configured, select **Test login**.
11. Once you've successfully logged in as both SuperAdmin and Organization Administrator, select **Finish** to complete the configuration.

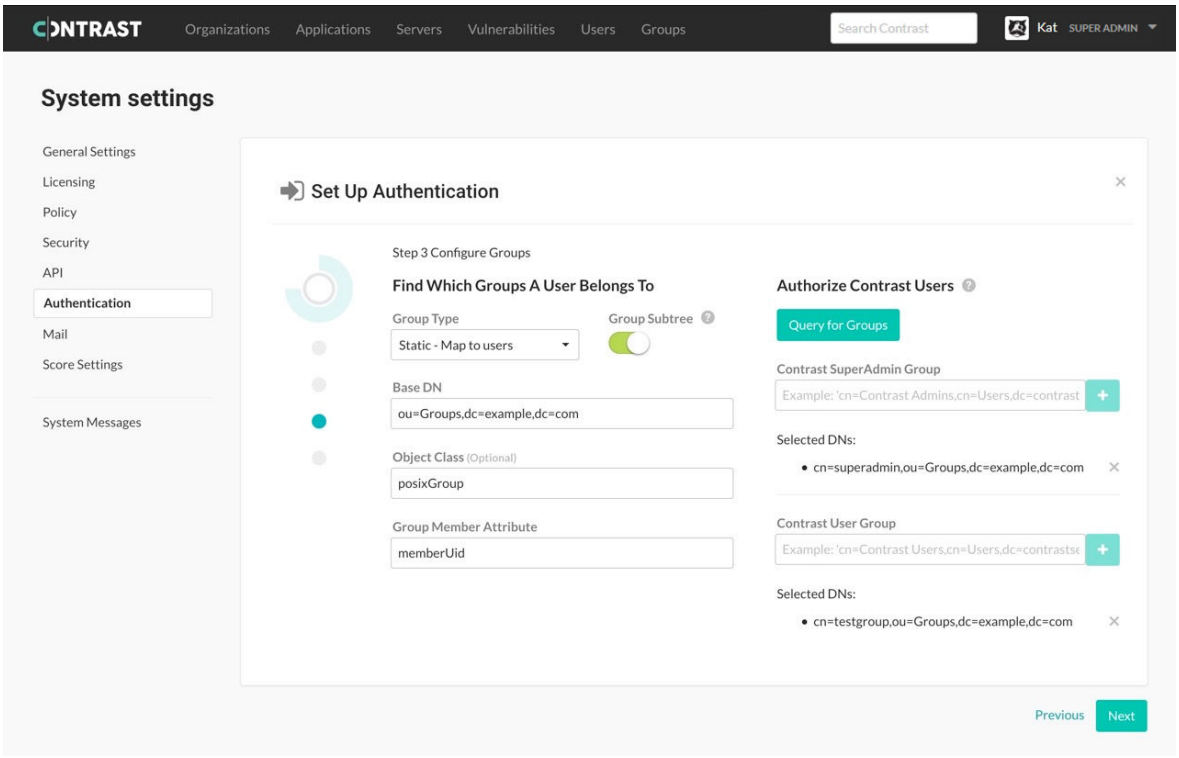
Configure groups for LDAP

As part of the LDAP configuration (page 717), you will need to configure groups.

Organization Administrators set the roles and permissions (page 742) 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 721). 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:

1. Enter the following values:



Option	Description	Default
Group type	Groups types depend on your server functionality and configuration. Groups are either: • **Static:** Groups track members through an attribute on the object, such as `uniqueMember`. The remaining four options in this table only apply to static groups. • **Dynamic:** The user object tracks its own membership. Groups are added dynamically to the user object when the user becomes a member of a group.	**Static**
Group subtree	Configures whether subtrees of the Base DN should be included when searching for groups in the directory.	**Enabled**
Base DN	This is the distinguished name (DN) where the application can find groups in your LDAP server (like the User Base DN).	`ou=Groups`
Object class	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
Group member attribute	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. 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"). If you use an RDN, Contrast does not see that user in the LDAP group.	`uniqueMember`

2. 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.

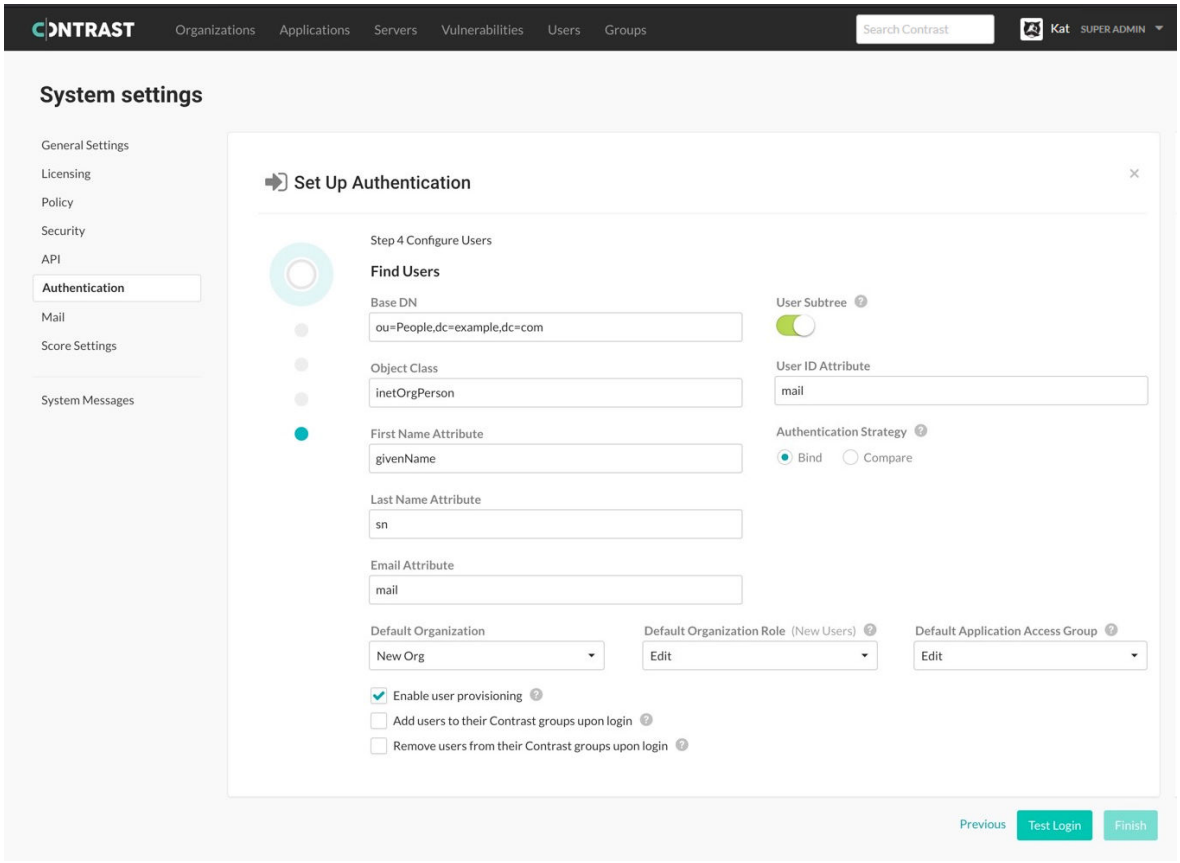
3. Select **Query for groups** to enable a live search of existing groups as you begin to type within the input fields.

Configure users for LDAP

As part of the LDAP configuration (page 717), 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.



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, *ou=Users*), but your LDAP administrator should verify that you're searching the right container.	`ou=users`
Object Class	Indicate the LDAP object class for user objects in the directory.	`inetOrgPerson`
First Name Attribute	Indicate the LDAP field that contains a user's first name.	`givenName`
Last Name Attribute	Indicate the LDAP field that contains a user's last name.	`sn`
Email Attribute	Indicate the LDAP field that contains a user's email address.	`mail`
User subtree	If enabled, subtrees of the Base DN are included when searching for users.	**Enabled**
User ID attribute	Indicate the LDAP field that should be identified as the User ID. This is the username to enter when logging in to contrast.	`uid`
Authentication strategy	Choose how you want Contrast to authenticate users when they provide their credentials. **Bind** means the application sends the user's credentials to the server for authentication. **Compare** means the server hashes the user's credentials and compares them to the value of the password attribute.	**Bind**

Option	Description	Default
Password attribute	The LDAP field that contains a user's password. If you selected **Compare** for the authentication strategy, this attribute contains the hashed password for the user.	`userPassword`

2. 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.
3. 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.

4. 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 717) 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. 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.
2. 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
```

3. 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.

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                             NameFormat="urn:oasis:names:tc:SAML:2.0:attrna
me-format:unspecified"
3                             >
4                <saml2:AttributeValue xmlns:xs="http://www.w3.org/2001/
XMLSchema"
5                                      xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance"
6                                      xsi:type="xs:string"
7                                      >Dan</saml2:AttributeValue>
8            </saml2:Attribute>
```

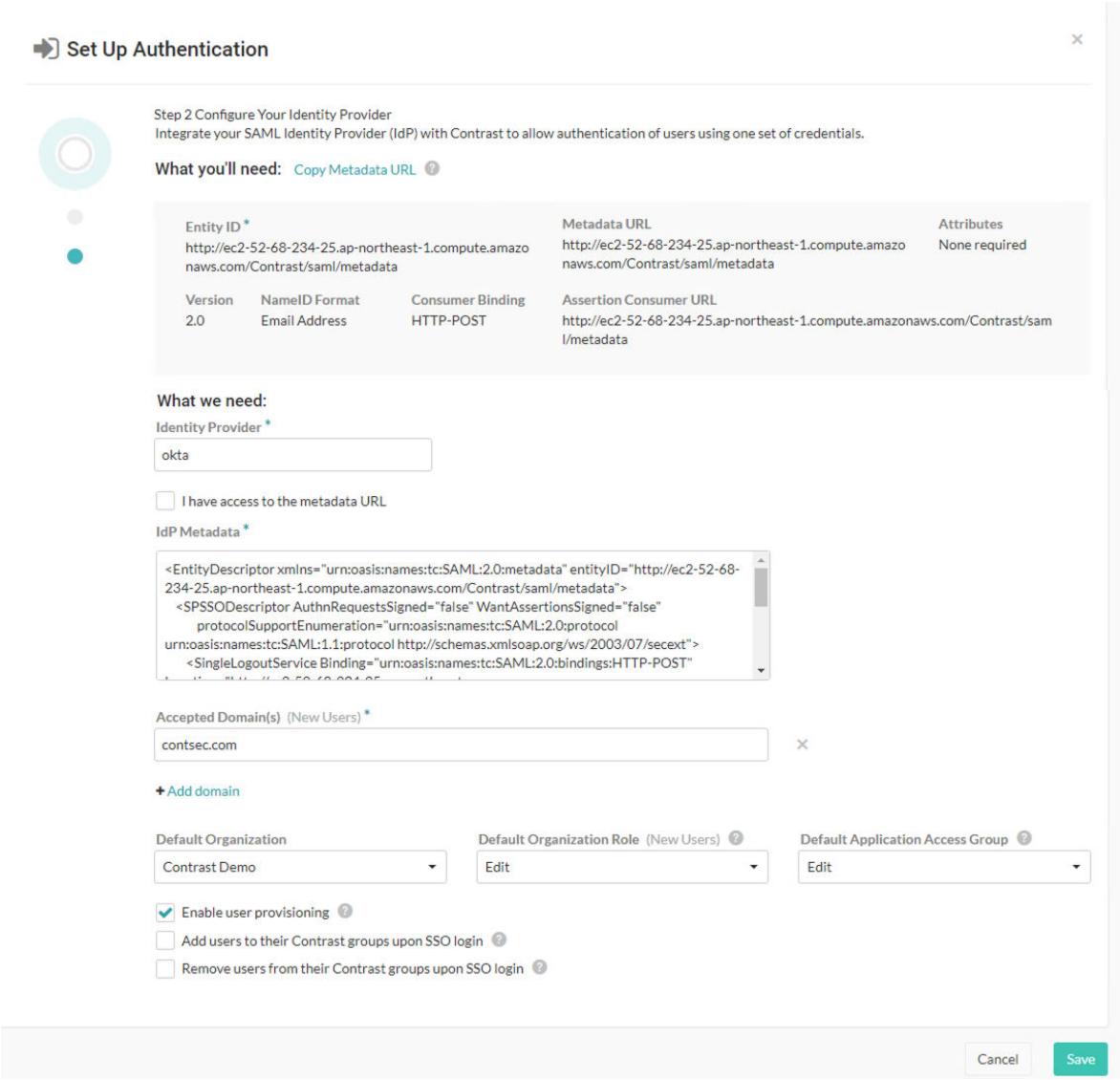
Steps

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 734) 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 712), 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.)



7. Provide a name for your **Identity provider**.
8. Enter your IDP metatdata. Select the box if you **have access to the metatdata URL**, then enter the URL.
9. 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 708).

10. Select **Save**. If an error occurs, you can check debug logs (page 691) for troubleshooting.
11. Restart Contrast (page 683) 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 683) 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 727), 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 639).
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 lockout**.
- Enter 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 days 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-sersver.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

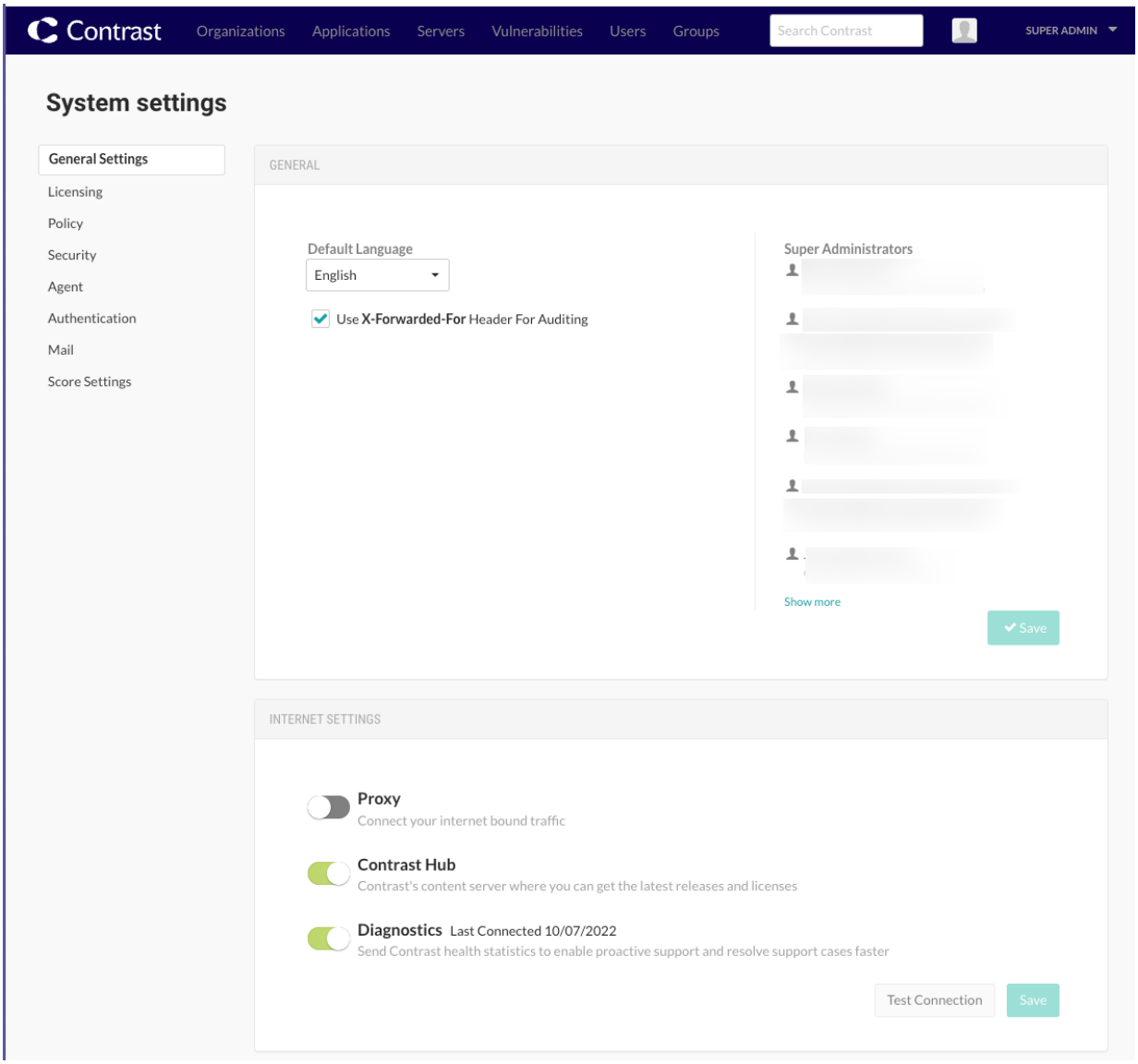
As a System Administrator, you can set policies for the keys across all your organizations to make sure they always meet your security standards.

1. In system settings (page 727), 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 32).
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 679) 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 729)
   - Licensing (page 730)
   - Policy (page 733) (library compliance settings)
   - Security (passwords (page 725), two-step authentication (page 436) and key management (page 727))
   - Agent keys (these are the same Agent keys (page 32) that you can find in Organization settings).
   - Authentication (page 712)
   - Mail (page 733)
   - Score settings (page 646)

Additional system settings

Additional system settings include:

- Configure Tomcat (page 685)
- Configure Java Runtime Environment (JRE) (page 686)
- Configure HTTPS (page 686)
- Add organizations (page 700)
- Configure log levels (page 692)
- Configure reporting storage (page 690)

Configure general system settings

General settings is part of system settings (page 727). 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. Due to privacy concerns, the data doesn't include application names, personally identifying information, code, vulnerability identities or customer network identifiers.

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 toggle to disable or enable it. Diagnostics are enabled by default.
3. Proxy settings apply to 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 673) and upgrade (page 694) Contrast.
- Hosted and on-premises customers require specific licenses for Assess (application licenses), Protect (server licenses), and SCA (library licenses).

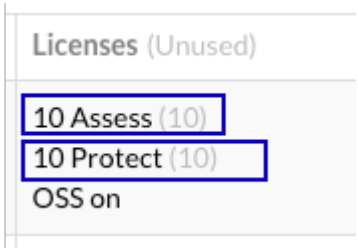
Use this procedure to allocate Assess and Protect licenses to organizations.

Before you begin

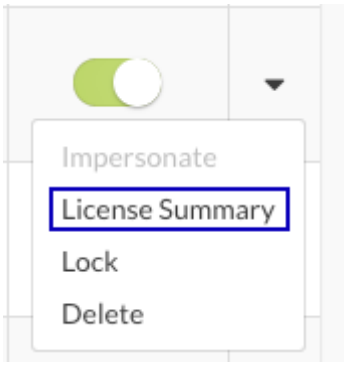
- For on-premises customers, a SuperAdmin or ServerAdmin role is required to allocate Assess and Protect licenses to a particular organization.
  Contrast Security handles this activity for hosted customers.
- Licenses applied to applications permanently count towards the number of maximum allowable applications. Deleting a licensed application has no effect on the number of licenses you are allowed to apply to applications.

Steps to allocate licenses

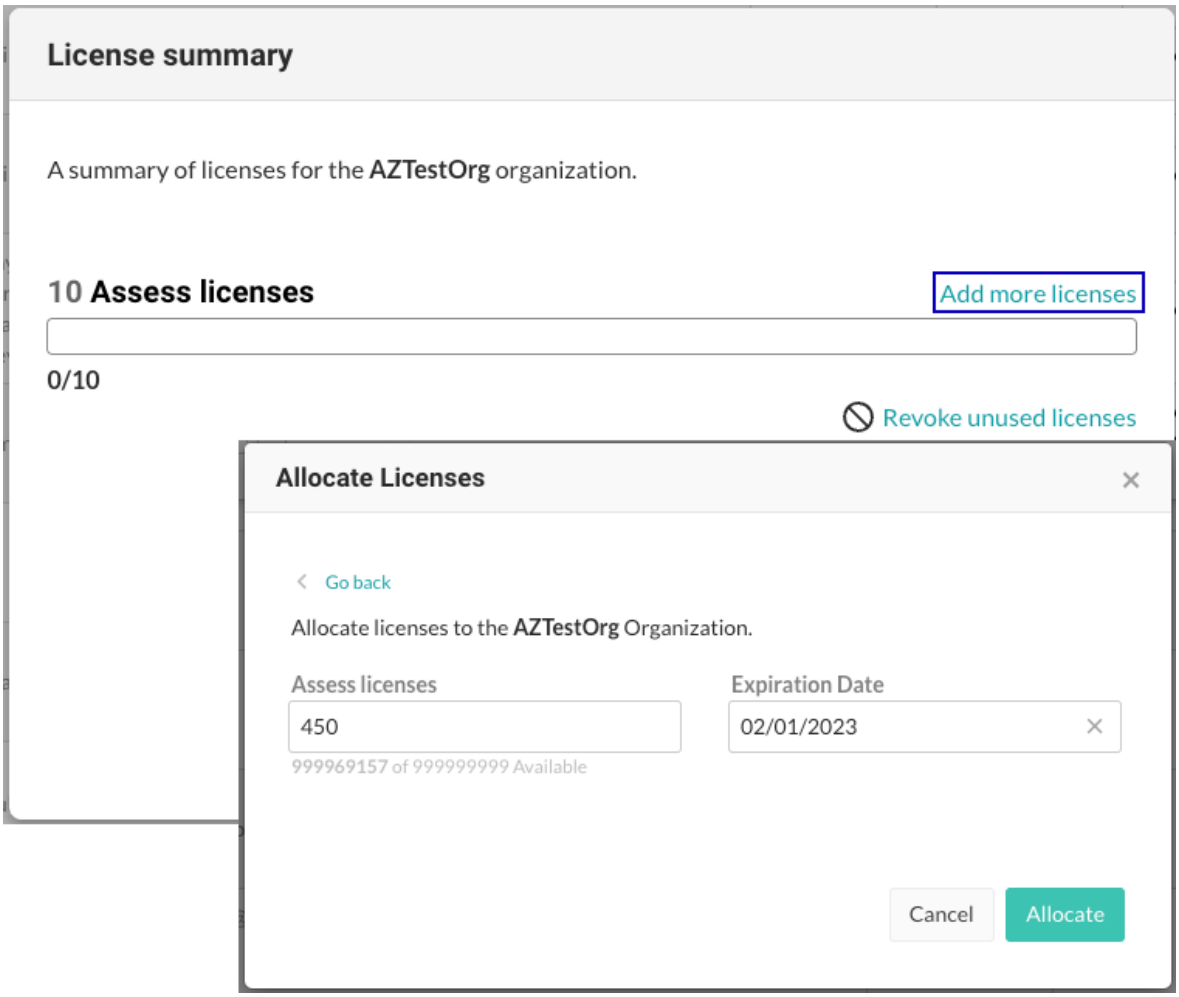
1. Select **SuperAdmin** or **ServerAdmin** in the **user menu**.
   You see a list of organizations. The **Licenses** column shows the total number of Assess and Protect licenses available for each organization, followed by the number of unused licenses in parentheses.
2. You see a list of organizations. 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.

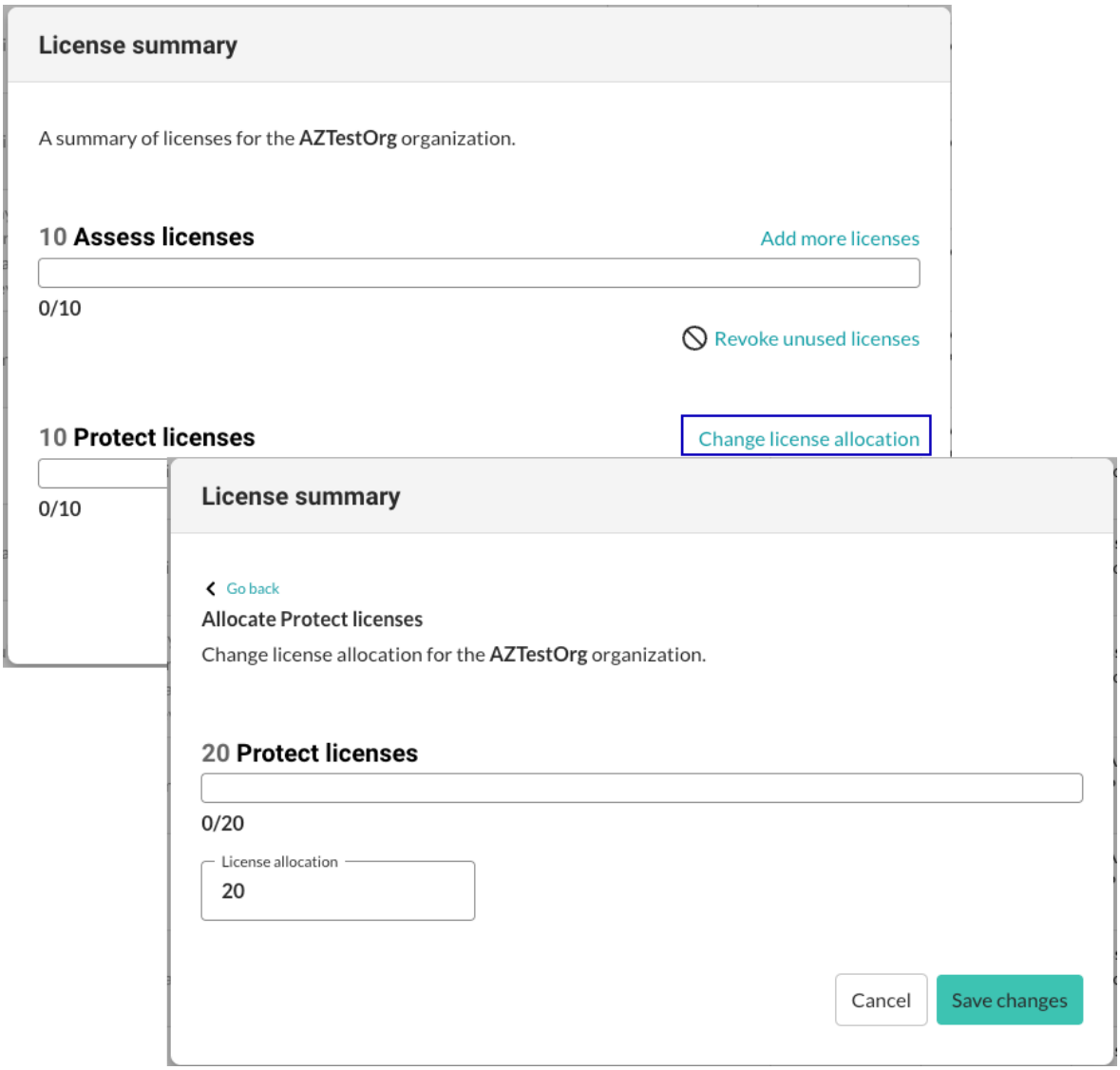
   - Select the triangle (˅) at the end of a row and select **License summary**.

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**.



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**.



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 700).
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 747), which can optionally depend on a library score (page 496). To customize score settings at a system level:

1. Under system settings (page 727), 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 496) 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 646)
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 727).
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 623) 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 643) for Contrast notifications at an organization level. Individual users can adjust their own settings (page 437).

To configure notifications at a system level:

1. Under system settings (page 727), 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 STARTTLSL:** 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 736)
- Improve performance (page 27)
- Upgrade Contrast (page 694)
- Upgrade the agents (page 695)
- Update library data (page 696)
- Update the Contrast license (page 698)
- Update the IP address (page 696)
- Manage SSL (page 736)
- Use the encrypted properties editor (page 734)

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
ad.properties	Settings to connect and configure Contrast to authenticate Active Directory groups.
ldap.properties	Settings to connect and configure Contrast to authenticate LDAP groups.
database.properties	Host and connection settings for communication between Contrast and MySQL.
saml.properties	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 682) 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.cmd`.
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.

Manage SSL

On-premises customers may need to use a Secure Sockets Layer (SSL) in the following situations:

- Setting up HTTPS proxy authentication (page 725)
- Integrating with LDAP (page 717) or Active Directory (page 713)
- Securing communication between agents and the Contrast application

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 736)
- Backup MySQL manually (page 736)
- Restore database backups (page 737)
- Disable automated backups (page 737)

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 *$CONTRAST_HOME/data/conf/database.properties* file with the encrypted editor (page 734).
2. Run the backup command for your environment:
   - **Windows:** `$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 673), 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 734) 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 734) 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.

Reference

These topics may be useful as an occasional reference on how to use Contrast:

- Glossary (page 739)
- Roles and permissions (page 742)
- Agent supported technologies (Java (page 52), .NET Framework (page 126), .NET Framework (Legacy) (page 171), .NET Core (page 181), Node.js (page 230), Python (page 295), Ruby (page 354), Go (page 410))
- Application scoring guide (page 747)
- Library scoring guide (page 496)
- Log levels (page 749)
- Supported browsers (page 753)
- Regular expressions for application exclusions (page 622)

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 attack (page 539) 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)	Common Vulnerabilities and Exposures is a list of publicly known cybersecurity vulnerabilities, used internationally to identify and track types of vulnerabilities.
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 dependency tree (page 495) in Contrast.
Contrast Hub	The Contrast Hub 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.
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 configure the agents (page 30) that instrument your applications and make sure they work within your preferred frameworks as expected and report metadata you want to see in Contrast.
false positive	A vulnerability that is falsely reported.
flow map	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.
instrument	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.
interactive application security testing (IAST)	Security technology that analyzes data flows within a running application to detect and report possible security vulnerabilities.
IP allowlist	An IP allowlist is a rule that allows any HTTP request from IP addresses on that list.

IP denylist	An IP denylist is a rule that blocks any HTTP request from IP addresses on that list.
library	A library is any packaged code included in an application. Libraries (page 488) can be public or private.
lightweight directory access protocol (LDAP)	LDAP is a lightweight client-server protocol for accessing and maintaining directory services. In Contrast, on-premises can use LDAP (page 717) to manage users and logins.
National Institute of Standards and Technology (NIST)	NIST is a government agency that promotes U.S. innovation and industrial competitiveness by advancing measurement science, standards, and technology, in multiple fields including cybersecurity.
Not a problem	*For library status*. This library has vulnerabilities that are acknowledged and the risks are acceptable.
Open Web Application Security Project (OWASP)	Open Web Application Security Project® 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 OWASP Top Ten lists the most critical security defects in Web applications.
path traversal	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.
policy	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.
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	*For library status.* The vulnerable library has been remediated.
Reported	*For library status*. Default status when a library with vulnerabilities is detected by Contrast.
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 library (page 496) and application (page 747) 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 data flow analysis (page 533), 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 data flow analysis (page 533), 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 (for example).
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.
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.

Roles and permissions

Permissions and capabilities are granted to users depending on the role they are assigned. Roles exist at the application (page 743), organization (page 744) and (for on-premises customers) system (page 746) levels. Most of these roles are defined when a user is assigned to an access group (page 636).

You can also:

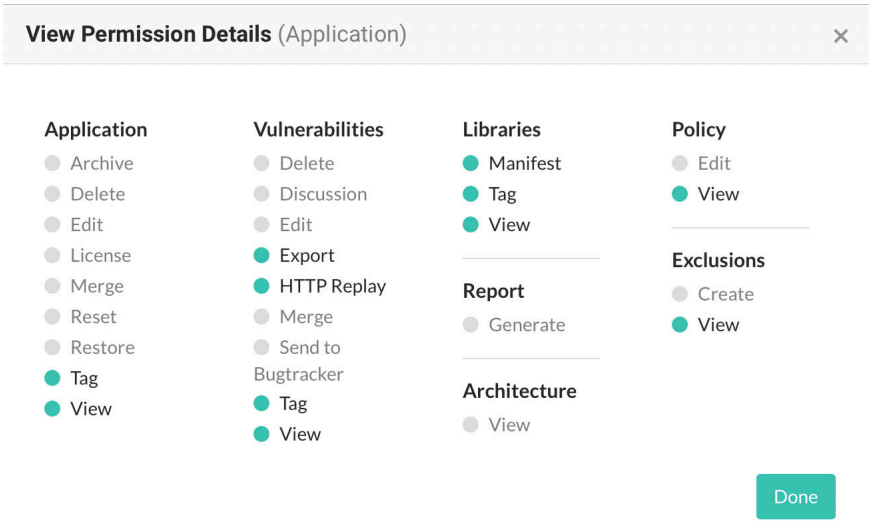
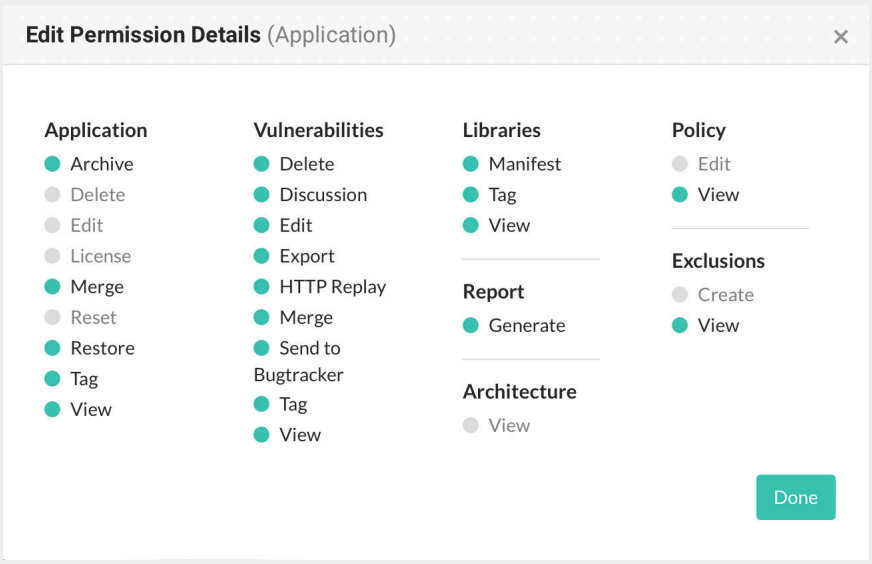
- View your own permissions (page 437)
- Manage organization permissions (page 635)
- Manage system permissions (page 699)
- Grant Protect permissions (page 707)

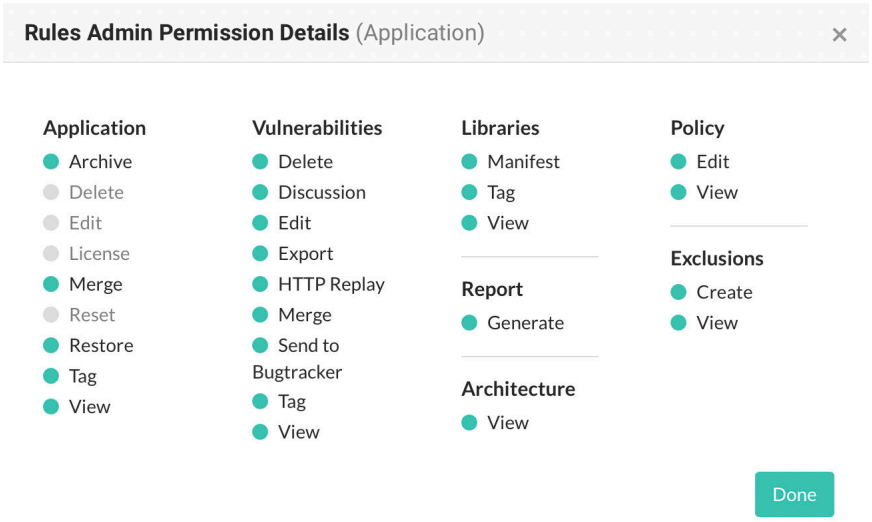
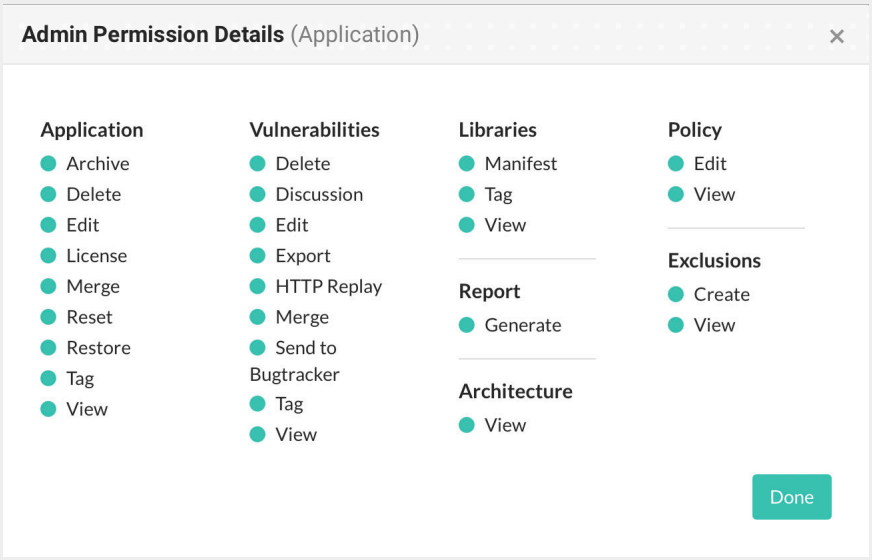
NOTE
API-Only users can access Contrast's REST API, but can't log in to the user interface. Contrast doesn't recommend the creation of administrator API accounts.

Application roles

Application roles give users permissions and capabilities within a particular application. Application roles are assigned to access groups (page 636).

Use these application roles to grant permissions and capabilities within an application:

View	The View role for applications (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. 
Edit	The Edit role for applications (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. 

Rules Admin	The Rules Admin role for applications (Application Rules Admin) can edit rules and policies in the application, enable Protect, and manage notifications and scoring for the application. 
Admin	The Admin role for applications (Application Administrator) has no restrictions and can manage other users' access to the application. 

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 636).

See also

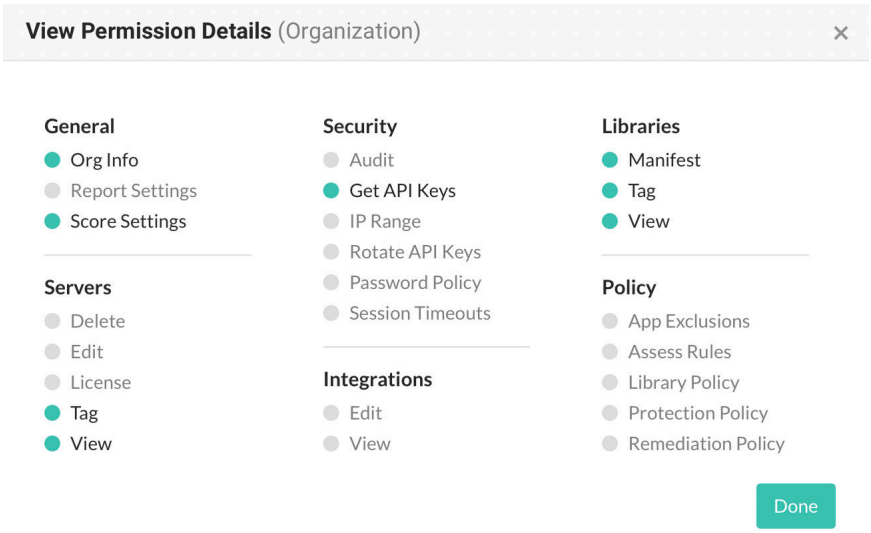
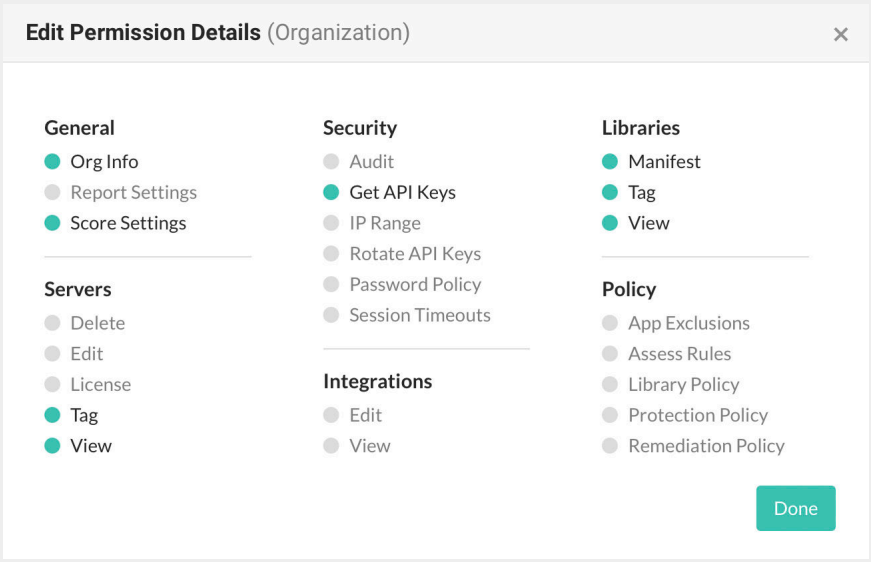
View permissions (page 437)

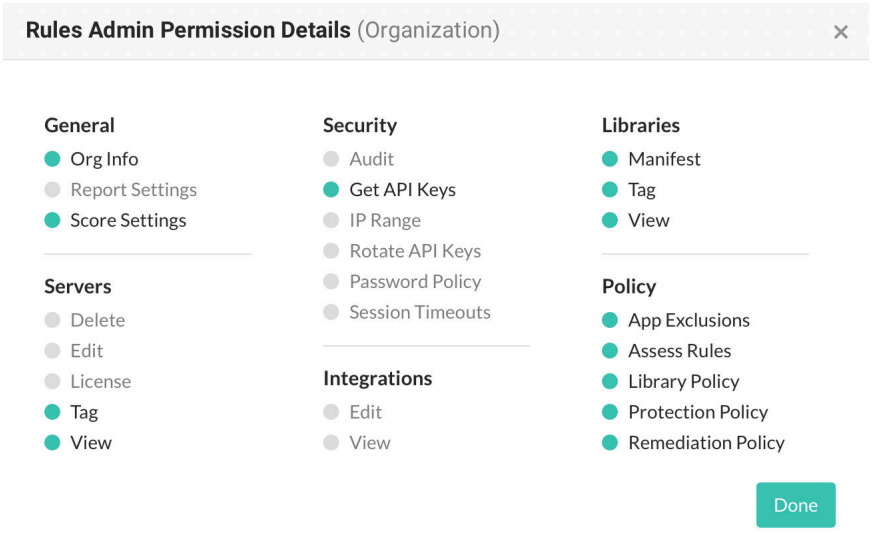
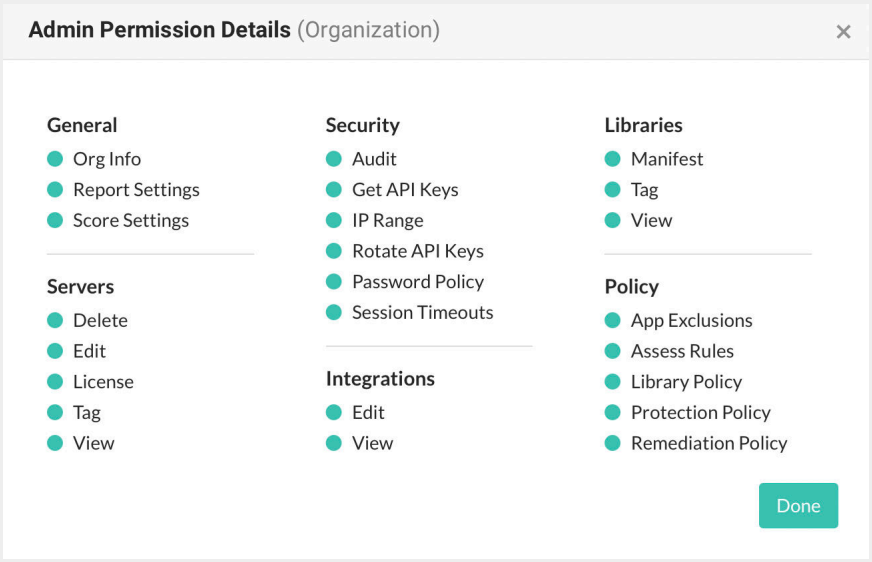
Organization roles

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 perform edits to traces to the application. 
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. 

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. 
Admin	The Admin role for an organization (Organization Administrator) is responsible for the configuration and management of the organization. 

You assign organization roles by adding users to an organization access group (page 636).

See also

View permissions (page 437)

System roles

NOTE

These roles are only available to on-premises customers.

When deciding how to manage system administration (page 699), 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 496) 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 X 0) - (10 X 1) - (5 X 2) - (1 X 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 609) 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 646) 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 x 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 x 10 = 20
```

If you have a library with the scores 2.4 and 2.2, the penalty would be:

```
2.4 x 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 692) or for a particular server (page 484) 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.
TRACE	Gives more detailed information than the DEBUG level.

Events and generic webhook variables

You can customize your generic webhook (page 573) response with data from Contrast events such as `NEW_VULNERABILITY` and `SERVER_OFFLINE`. Each event contains general (page 575), application (page 575), server (page 575) or vulnerability (page 575) variables you can call in your payload request.

Event	Variables
`ATTACK_END`	General (page 575), Application (page 575), Server (page 575)
`ATTACK_EVENT_COMMENT`	General (page 575), Application (page 575), Server (page 575)

Event	Variables
ATTACK_UPDATE	General (page 575), Application (page 575), Server (page 575)
EXPIRING_LICENSE	General (page 575), Application (page 575)
NEW_ASSET (if new application)	General (page 575), Application (page 575) and Server (page 575) (if new application)
NEW_ATTACK_APPLICATION	General (page 575), Application (page 575), Server (page 575)
NEW_ATTACK_UPDATE	General (page 575), Application (page 575), Server (page 575)
NEW_ATTACK	General (page 575), Application (page 575), Server (page 575)
NEW_VULNERABILITY_COMMENT	General (page 575), Application (page 575), Server (page 575), Vulnerability (page 575)
NEW_VULNERABILITY	General (page 575), Application (page 575), Server (page 575), Vulnerability (page 575)
NEW_VULNERABLE_LIBRARY	General (page 575), Application (page 575)
SERVER_OFFLINE	General (page 575), Server (page 575)
VULNERABILITY_CHANGESTATUS_CLOSED	General (page 575), Application (page 575), Server (page 575), Vulnerability (page 575)
VULNERABILITY_CHANGESTATUS_OPEN	General (page 575), Application (page 575), Server (page 575), Vulnerability (page 575)
VULNERABILITY_DUPLICATE	General (page 575), Application (page 575), Server (page 575), Vulnerability (page 575)

Generic webhook variables

You can customize your generic webhook (page 573) response with data from Contrast events such as NEW_VULNERABILITYand 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
General variables	
$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”
Application variables	
$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

Variables	Description
`$ApplicationLastSeen`	When the application was last seen, in Unix time For example: 1572033840000
`$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.
`$ApplicationTotalModules`	The number of modules your application has
Server variables	
`$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
Vulnerability variables	
`$Severity`	If this event is triggered by a vulnerability, this is the severity of the vulnerability
`$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
`$VulnerabilityDefaultSeverity`	Default severity of the vulnerability
`$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

Variables	Description
`$VulnerabilityInstanceLastTimeSeen`	Last time the vulnerability was seen, in Unix time For example: 1572033840000
`$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 620):

Effect	Pattern	Example pattern	Example match
Start of a string	`^`	`^w+`	`Start` of a string
End of a string	`$`	`w+$`	End of a `string`
A single character of: a, b or c	`[abc]`	`[abc]+`	a bb ccc
A character except: a, b or c	`[^abc]`	`[^abc]+`	`Anything`but`abc.`
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]+`	Anything`buta-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
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.

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 729)
- .NET Framework and .NET Core agent telemetry (page 179)
- Ruby telemetry (page 409)
- Python telemetry (page 352)

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 for investigation.