IQ Server

Table of Contents

The Nexus IQ Server is the policy engine that powers Nexus Firewall, Lifecycle, and Auditor. With IQ Server, you can do the following:

- Share component intelligence with your teams so they make better decisions and build better software.
- Implement a fully-customizeable policy engine letting you define which components are acceptable and which are not.
- Integrate with popular development tools like Maven, Eclipse, IntelliJ, Visual Studio, GitHub, Bamboo, Jenkins, Xebia Labs, and SonarQube.
- Provide a full suite of supported REST APIs (see page 235) that provide access to core features for custom implementations.

Our documentation is written to match the latest available release of the IQ Server and any associated integrations[1].

Navigate IQ Server Help using the page tree in the left sidebar:

To see the latest changes and updates to the IQ Server, please see the Release Notes (see page 13). To download the latest version, see Download and Compatibility (see page 53).

As always, if you need any further assistance, please don't hesitate to consult https://support.sonatype.com/home or contact us at support@sonatype.com[2].

Release Notes

We're continuously improving Nexus IQ Server products and features based on customer feedback. We make a lot of enhancements regularly, and our release notes provide detailed descriptions of each product release with links to additional technical information and support resources.

As a best practice, we recommend that you keep your IQ Server installation up to date so you can benefit from the latest features and advancements in component intelligence. The latest version can be downloaded from the IQ Download and Compatibility (see page 53) page.

If you are upgrading from an earlier version of IQ Server, please see Upgrading the IQ Server (see page 90).

1 https://help.sonatype.com/display/NXI/Nexus+Integrations
2 mailto:support@sonatype.com

Coming Soon

Improved JavaScript Reporting

We will be streamlining the display of JavaScript results by significantly reducing the noise. Individual JavaScript files identified as belonging to the same component will be aggregated and presented as a single line item. This will improve the readability and comprehensibility of JavaScript results while retaining all of the discovered vulnerabilities and occurrences. More information on the impact of these changes can be found in our documentation (see page 227).

1.49 (July 2018)

Security Fixes for HTTP Connector (Jetty)

Updated Jetty to fix several vulnerabilities related to HTTP request parsing.

1.48 (June 2018)

Policy-centric Component Information Panel

The Component Information Panel (see page 206) has been updated to display policy violations instead of security vulnerabilities and license issues.

Fixes for Recording of Component Occurrences

In some cases, the IQ CLI and our CI plugins for Bamboo and Jenkins recorded incomplete pathnames for the components in applications, causing misleading information in the *Occurrences* tab and issues in detecting proprietary components. We have fixed this in the new versions of those tools but there's a catch: If you previously configured regular expressions for proprietary components that only match exact pathnames, i.e. do not start with `.*` or similar wildcards, those regular expressions might need updating to account for the fixed pathnames.

Security Fix for JavaMail

Updated JavaMail to fix leaking host and username in message headers (SONATYPE-2017-0492).

Security Fix for Lifecycle XC CLI

Updated a vulnerable dependency of Nexus IQ CLI to fix a Zip Slip vulnerability that was exposed when running in its XC (Expanded Coverage) (see page 80) mode.

Component Labels REST API

Nexus IQ Server now has a Component Labels REST API (see page 276) for adding and removing component labels for an application.

1.47 (April 2018)

Component Versions REST API

Nexus IQ Server now has a Component Versions REST API (see page 274) for returning a list of versions for a component.

Improvements to Getting Started Page

The "Getting Started" page now indicates if there are any connectivity issues with Sonatype Data Services.

Persistent Warning to Change Default Password

A persistent warning is displayed if the default password for the built-in 'admin' account is not changed.

Automatic Application Creation for Nexus IQ for Jenkins

Nexus IQ for Jenkins 3.0.20180425-130011.728733c now supports automatic application creation (see page 144).

1.46 (April 2018)

Automatic Application Creation for Sonatype CLM for Maven

Sonatype CLM for Maven 2.8.1-01 now supports automatic application creation (see page 144). As part of these changes, anonymous access is no longer supported and credentials must be provided in order to communicate with Nexus IQ Server.

Automatic Application Creation for Nexus IQ for Bamboo

Nexus IQ for Bamboo 1.8.0 now supports automatic application creation (see page 144).

RubyGems Data Available in Nexus Firewall

RubyGems packages are now supported in Nexus Firewall. Available data includes: identification, licenses, and security vulnerabilities.

Getting Started Page

Nexus IQ Server has added a "Getting Started" page to facilitate onboarding administrative users. For non-administrative users a list of helpful "Learning Topics" are provided.

Performance Fix

A performance issue was found in 1.45 with certain access patterns to violation data. This has been fixed in 1.46. All users of 1.45 are advised to upgrade.

1.45 (March 2018)

Improved Database Format for Reduced Disk Space Consumption

This version of Nexus IQ Server uses a revised format to store the policy violation data to reduce its disk space consumption. Especially installations that have applications with a long history of policy evaluations or with a high frequency of policy evaluations will benefit from this upgrade.

⚠ Depending on the size of your existing installation and the hardware running your IQ Server, upgrading to this new version can take notable time. Be sure to read the instructions for Upgrading the IQ Server to Version 1.45 (see page 93) to prepare yourself appropriately.

Automatic Application Creation

Nexus IQ Server now allows automatic creation of applications (see page 144). Users with permission to manage automatic application creation can enable this feature and specify the parent organization for any automatically-created applications. When enabled, if a policy evaluation is performed for an application ID that does not exist, a new application with that ID will be created automatically instead of failing. Only the Nexus IQ CLI has been updated to take advantage of this new feature as of this release.

Anonymous Access Removed in Nexus IQ CLI

Nexus IQ CLI no longer supports anonymous access. With this change we begin the process of phasing out support for anonymous access from Nexus IQ clients.

1.44 (February 2018)

Login Modal Styling Improvements

The Nexus IQ Server login window has been updated with styling that matches the other forms within the application.

Automatic Import of Reference Policies

Upon first start, the Nexus IQ Server will now automatically download and import the current Reference Policy Set (see page 146). This removes the need for an administrator to manually find, download, and import policies when getting started with IQ for the first time. The manual import capability is still provided.

1.43 (January 2018)

Configuration Changes due to Upgraded Server Infrastructure

The Nexus IQ Server infrastructure has been upgraded, bringing with it many benefits including a more powerful configuration format for its networking and logging.

⚠ If you wish to use a configuration file from a prior version, then you must update it. Please refer to our configuration update guide (see page 107) for more information.

Product License Page Improvements

The Product License page has been enhanced to display additional important information including company name, primary contact name and e-mail address, license type(s), licensed users, expiration date, and days remaining. Additionally, we have provided more guidance for license installations.

Sandbox Organization and Application for Fresh Installs

Fresh installations starting with this version will, by default, create a "Sandbox Organization" with a child "Sandbox Application". This is to help facilitate the training of new users by providing a premade and safe sandbox for them to learn within. Please refer to the sample data configuration (see page 106) for more information.

Other Versions

IQ Server release notes are organized by year:

- 2018 Release Notes (see page 18) (1.43 and up)
- 2017 Release Notes (see page 23) (1.25 - 1.42)
- 2016 Release Notes (see page 33) (1.19 - 1.24)
- 2015 Release Notes (see page 41) (1.13 - 1.18)
- 2014 Release Notes (see page 48) (1.12)

2018 Release Notes

⚠ Sonatype encourages using the most current IQ Server release and not trailing behind more than six months. Release notes for the most current versions can be viewed here (see page 13) .

Coming Soon

Improved JavaScript Reporting

We will be streamlining the display of JavaScript results by significantly reducing the noise. Individual JavaScript files identified as belonging to the same component will be aggregated and presented as a single

line item. This will improve the readability and comprehensibility of JavaScript results while retaining all of the discovered vulnerabilities and occurrences. More information on the impact of these changes can be found in our documentation (see page 227).

1.49 (July 2018)

Security Fixes for HTTP Connector (Jetty)

Updated Jetty to fix several vulnerabilities related to HTTP request parsing.

1.48 (June 2018)

Policy-centric Component Information Panel

The Component Information Panel (see page 206) has been updated to display policy violations instead of security vulnerabilities and license issues.

Fixes for Recording of Component Occurrences

In some cases, the IQ CLI and our CI plugins for Bamboo and Jenkins recorded incomplete pathnames for the components in applications, causing misleading information in the *Occurrences* tab and issues in detecting proprietary components. We have fixed this in the new versions of those tools but there's a catch: If you previously configured regular expressions for proprietary components that only match exact pathnames, i.e. do not start with `.*` or similar wildcards, those regular expressions might need updating to account for the fixed pathnames.

Security Fix for JavaMail

Updated JavaMail to fix leaking host and username in message headers (SONATYPE-2017-0492).

Security Fix for Lifecycle XC CLI

Updated a vulnerable dependency of Nexus IQ CLI to fix a Zip Slip vulnerability that was exposed when running in its XC (Expanded Coverage) (see page 80) mode.

Component Labels REST API

Nexus IQ Server now has a Component Labels REST API (see page 276) for adding and removing component labels for an application.

1.47 (April 2018)

Component Versions REST API

Nexus IQ Server now has a Component Versions REST API (see page 274) for returning a list of versions for a component.

Improvements to Getting Started Page

The "Getting Started" page now indicates if there are any connectivity issues with Sonatype Data Services.

Persistent Warning to Change Default Password

A persistent warning is displayed if the default password for the built-in 'admin' account is not changed.

Automatic Application Creation for Nexus IQ for Jenkins

Nexus IQ for Jenkins 3.0.20180425-130011.728733c now supports automatic application creation (see page 144).

1.46 (April 2018)

Automatic Application Creation for Sonatype CLM for Maven

Sonatype CLM for Maven 2.8.1-01 now supports automatic application creation (see page 144). As part of these changes, anonymous access is no longer supported and credentials must be provided in order to communicate with Nexus IQ Server.

Automatic Application Creation for Nexus IQ for Bamboo

Nexus IQ for Bamboo 1.8.0 now supports automatic application creation (see page 144).

RubyGems Data Available in Nexus Firewall

RubyGems packages are now supported in Nexus Firewall. Available data includes: identification, licenses, and security vulnerabilities.

Getting Started Page

Nexus IQ Server has added a "Getting Started" page to facilitate onboarding administrative users. For non-administrative users a list of helpful "Learning Topics" are provided.

Performance Fix

A performance issue was found in 1.45 with certain access patterns to violation data. This has been fixed in 1.46. All users of 1.45 are advised to upgrade.

1.45 (March 2018)

Improved Database Format for Reduced Disk Space Consumption

This version of Nexus IQ Server uses a revised format to store the policy violation data to reduce its disk space consumption. Especially installations that have applications with a long history of policy evaluations or with a high frequency of policy evaluations will benefit from this upgrade.

⚠ Depending on the size of your existing installation and the hardware running your IQ Server, upgrading to this new version can take notable time. Be sure to read the instructions for Upgrading the IQ Server to Version 1.45 (see page 93) to prepare yourself appropriately.

Automatic Application Creation

Nexus IQ Server now allows automatic creation of applications (see page 144). Users with permission to manage automatic application creation can enable this feature and specify the parent organization for any automatically-created applications. When enabled, if a policy evaluation is performed for an application ID that does not exist, a new application with that ID will be created automatically instead of failing. Only the Nexus IQ CLI has been updated to take advantage of this new feature as of this release.

Anonymous Access Removed in Nexus IQ CLI

Nexus IQ CLI no longer supports anonymous access. With this change we begin the process of phasing out support for anonymous access from Nexus IQ clients.

1.44 (February 2018)

Login Modal Styling Improvements

The Nexus IQ Server login window has been updated with styling that matches the other forms within the application.

Automatic Import of Reference Policies

Upon first start, the Nexus IQ Server will now automatically download and import the current Reference Policy Set (see page 146). This removes the need for an administrator to manually find, download, and import policies when getting started with IQ for the first time. The manual import capability is still provided.

1.43 (January 2018)

Configuration Changes due to Upgraded Server Infrastructure

The Nexus IQ Server infrastructure has been upgraded, bringing with it many benefits including a more powerful configuration format for its networking and logging.

⚠️ If you wish to use a configuration file from a prior version, then you must update it. Please refer to our configuration update guide (see page 107) for more information.

Product License Page Improvements

The Product License page has been enhanced to display additional important information including company name, primary contact name and e-mail address, license type(s), licensed users, expiration date, and days remaining. Additionally, we have provided more guidance for license installations.

Sandbox Organization and Application for Fresh Installs

Fresh installations starting with this version will, by default, create a "Sandbox Organization" with a child "Sandbox Application". This is to help facilitate the training of new users by providing a premade and safe sandbox for them to learn within. Please refer to the sample data configuration (see page 106) for more information.

2017 Release Notes

⚠ Sonatype encourages using the most current IQ Server release and not trailing behind more than six months. Release notes for the most current versions can be viewed here (see page 13) .

1.42 (December)

Java 8 is now required for Nexus IQ Server and Nexus IQ CLI

Oracle Java 8 is now required in order to run Nexus IQ Server and Nexus IQ CLI (previously Java 7 was supported). Using prior versions of Java will fail with an `UnsupportedClassVersionError` on startup.

Organization REST API now supports creating organizations

The Organization REST API has been enhanced to allow creating new organizations via POST operations. The Organization REST API (see page 268) documentation provides more details on usage.

Nexus IQ CLI now displays scan fingerprinting performance information

The Nexus IQ CLI now displays the number of archives and files fingerprinted and the duration (in seconds) required to fingerprint them.

Nexus IQ Server `forceBaseUrl` option now works correctly with reverse proxies that change the web application context path

Nexus IQ Server 1.41 failed to load fonts and icons when running behind a reverse proxy that changed the web application context path. The `forceBaseUrl` option now works as intended in such circumstances.

1.41 (November)

Force baseUrl for user-facing URLs

Sonatype encourages the use of the `X-Forwarded-Proto` and `X-Forwarded-Host` headers set by proxies to match a user-facing URL. If you are not able to use these headers then please contact our customer support team[3] for assistance on how to force the use of the baseUrl.

⚠ The ability to force the baseUrl will be removed in 6 months.

Support for evaluating Java 9 applications and components

The application and component evaluation have been updated to support Java 9 bytecode.

Support for the new Twistlock Console 2.2 in Nexus IQ CLI

We added support for the new Twistlock CLI tool. The new integration requires at least Twistlock 2.2.100. The old `twistlock-scanner` Twistlock CLI tool is not supported anymore. If you cannot update to at least Twistlock 2.2.100, you can still use previous versions of Nexus IQ CLI.

1.40 (November)

Removal of Calculate Trends

The *Calculate Trends* feature previously available in the IQ *Dashboard* has been removed. Please use Success Metrics (see page 230) for similar functionality.

Configuring baseUrl no longer forces the application URL

Setting the baseUrl will only change the application URL for email notifications. Requests from any application URL through IQ Server will be maintained. Specifically, proxies should make use of the `X-Forwarded-Proto` and `X-Forwarded-Host` headers to match a user-facing URL.

1.39 (October)

The IQ Server 1.39 release contains the following updates:

3 mailto:support@sonatype.com

Success Metrics Performance

Repeated loads of a Success Metrics Report within the same aggregation interval will now load more quickly due to additional data caching within the IQ Server.

Always up-to-date Success Metrics

Success Metrics Reports can now be configured to update constantly. Previously, the reports would only update at the beginning of each calendar month. When creating a Success Metrics Report, the user may now choose the aggregation interval for the report, with options of "by calendar month" and "by most recent evaluation." This replaces the "daily aggregations" feature that was added to Success Metrics in the 1.36 release of IQ Server. See the Success Metrics (see page 230) page for more details.

New Save Filter workflow

The *Save Filter* modal on the IQ *Dashboard* has been updated to help the user more clearly distinguish between overwriting an existing filter and saving a filter under a new name.

Improved Sorting of Dashboard Results

Manually sorting columns within the IQ *Dashboard* will now sort over the entire result set instead of only the limited number of rows that are displayed within the browser.

Security Fixes for IQ plugins

New versions of Maven, Hudson/Jenkins 1.x, and SonarQube plugins have been released to resolve vulnerable, but not exploitable, dependencies.

1.38 (October)

The IQ Server 1.38 release contains the following update:

Lifecycle XC Report Improvements

Component names in the *Identified Components* section of Lifecycle XC reports have been standardized across several of the supported ecosystems and formats. This improvement aims to bring further clarity to the user in recognizing the reported components. For more information, please see the Lifecycle XC (see page 80) topic.

1.37 (September)

The IQ Server 1.37 release contains the following updates:

Success Metrics reports for selected Organizations and Applications

Users can now configure Success Metrics (see page 230) reports for a specified scope made up of a selection of organizations and applications. The user may either specify exactly what they want, or select an option to include all applications. Each user can now have multiple Success Metrics reports, whereas previously only a report for the entire Root Organization was available.

1.36 (September)

The IQ Server 1.36 release contains the following updates:

Security Vulnerability Presence Policy Condition Replaced with Security Vulnerability Severity Policy Condition

In an effort to reduce redundant policy conditions, the *Security Vulnerability present* policy condition has been replaced by the *Security Vulnerability Severity greater than or equal to 0* policy condition. Additionally, the *Security Vulnerability absent* policy condition has been removed. If any of your policies use the *Security Vulnerability absent* policy condition, then please see our upgrade instructions (see page 90) before upgrading, and contact our customer support team[4] or your customer success representative for assistance in changing them before upgrading to ensure a successful migration.

Success Metrics

Success Metrics (see page 230) has been updated to calculate aggregations daily for new installations where historical data is limited.

1.35 (August)

The IQ Server 1.35 release contains the following updates:

Lifecycle XC

Lifecycle XC (Expanded Coverage) is a new capability of Nexus Lifecycle that utilizes OWASP dependency-check[5] to provide basic coverage for additional languages. Specifically, Nexus IQ CLI features a new option

4 mailto:support@sonatype.com
5 https://jeremylong.github.io/DependencyCheck/index.html

to run in normal (Lifecycle) or XC mode. When XC mode is enabled Nexus IQ CLI will be configured to scan and analyse a different set of ecosystems and formats including Ruby, Swift, CocoaPods, and PHP. For more information, please see the Lifecycle XC (see page 80) topic.

Success Metrics Components tile

Success Metrics (see page 230) has been updated with a new Components tile that breaks down which components are used across the most applications and which components have the most policy violations.

1.34 (July)

The IQ Server 1.34 release features support for Docker image evaluations natively using Nexus IQ tooling. Updates to the latest version of tools are required for Docker image evaluations. This includes:

Nexus IQ Tool	Minimum Required Version
Nexus IQ CLI	1.34.0
Nexus IQ for Bamboo	1.5.1
Nexus IQ for Jenkins 2	1.3.20170728-122322.902d97e
Nexus IQ for Hudson/Jenkins 1	2.19.0

An update to the Nexus IQ Server is not required. More information on performing Docker image evaluations is available in the Evaluating an Application section of the Nexus IQ CLI[6] topic and the Docker Images section of Nexus IQ for Jenkins[7].

1.33 (July)

The IQ Server 1.33 release contains the following updates:

Support to Upgrade from Nexus Repository Manager 2 to 3 When Using Nexus Firewall

Nexus Repository Manager 3.5.0 enables you to upgrade from Nexus Repository Manager 2.14.5 and retain the state of any proxy repositories that use the audit or quarantine functionality of Nexus Firewall. This version of IQ Server is required to assist the upgrade process for those repositories using Nexus Firewall.

6 https://help.sonatype.com/display/NXI/Nexus+IQ+CLI
7 https://help.sonatype.com/display/NXI/Nexus+IQ+for+Jenkins

Success Metrics

In the toolbar you can now find a link to Success Metrics, which summarize important IQ Server activity over the last 12 months. The data is gathered for all organizations and applications you have access to. Please note that you need at least one month of historical evaluation data to generate Success Metrics. They can be disabled by the System Administrator via the System Preferences menu.

1.32 (July)

The IQ Server 1.32 release contains the following updates:

Nexus IQ CLI Supports Parameters from Files

Support for the Nexus IQ CLI to load parameters from files has been added. For more information, please see "Loading Parameters from a File" in the Nexus IQ CLI[8] topic.

Nexus IQ CLI Requires Java 7

Starting with this version, Nexus IQ CLI requires Java 7 (was Java 6 before).

Policy Coordinate Condition Update with Support for Extension and Classifier

The policy coordinate condition has been updated to allow the extension and classifier coordinates to be specified when creating a Maven coordinate condition for a policy constraint. Inputting coordinates has also been changed such that required coordinates must have a value specified whereas optional coordinates can be left empty. For Maven, only the classifier coordinate is optional, and for A-Name only the qualifier coordinate is optional. A coordinate with an empty value will only match a component if it does not have that coordinate. A coordinate with a wildcard value will match a component with any value for that coordinate as well as if it did not have that coordinate.

Uncategorized Applications Filter on Dashboard

The Application Categories Filter on the Dashboard now has a new option: "No Category". This option allows the user to explicitly control whether they want applications which are not a member of any category to appear in the Dashboard results. Previously, uncategorized applications would never be included when any specific category filters were selected.

8 https://help.sonatype.com/display/NXI/Nexus+IQ+CLI

Security Fix for CI Plugins

Both Nexus IQ for Bamboo and Nexus IQ for Hudson/Jenkins 1.x suffered from a vulnerability related to deserialization of XML files that could crash the JVM. This release provides updated versions of those two plugins to resolve the issue.

Updated Minimum Requirements for CI Plugins

To facilitate the aforementioned security fix, version 1.4.0 of Nexus IQ for Bamboo now requires at least Bamboo 5.10 as host runtime. This also implies a requirement on Java 8 as needed to run Bamboo.

Likewise, version 2.18.0 of Nexus IQ for Hudson/Jenkins 1.x requires Java 8 as runtime and versions of Hudson before 3.2.0 are no longer supported.

Please note that in the next few months, we plan to adopt Java 8 as the minimum runtime requirement for all other client integrations and IQ Server itself. If you have any questions or concerns regarding this, you can reach out to us via the IQ Feedback Group[9].

1.31 (June)

The IQ Server 1.31 release contains the following update:

IQ for Visual Studio

Support for the Nexus IQ Extension for Microsoft Visual Studio has been added to the IQ Server. The extension can be installed from within Microsoft Visual Studio using the Extensions manager or from the Visual Studio Marketplace.

1.30 (May)

The IQ Server 1.30 release contains the following updates:

Customizable System Notice

In the system preferences, system administrators can now configure a system notice. When enabled, the system notice will be displayed in the login dialog as well as on top of every page of the web UI.

9 mailto:iq-feedback-group@sonatype.com

Dashboard UI Improvements

The Violations tab on the Dashboard has been enhanced for easier readability. Results are now shown in a single row, and the Latest Report link has its own column for simple access. For more information, please see the Dashboard (see page 190) topic.

1.29 (May)

The IQ Server 1.29 release contains the following update:

Optional changes to dashboard filter defaults

This release includes an option to change the behavior of the initial dashboard state to alleviate performance concerns. Contact the product team if you're interested in trying this feature and providing feedback.

1.28 (May)

The IQ Server 1.28 release contains the following update:

PyPI data available in Nexus Firewall

PyPI packages are now supported in Nexus Firewall. Available data includes: identification, licenses, and security vulnerabilities.

1.27 (April)

The IQ Server 1.27 release contains the following updates:

Dashboard Filtering by Violation State

The Dashboard now allows you to filter its views by the state of a policy violation. That is, you can also include waived violations in your reviews of application health. Likewise, you can exclude all open violations and focus on the effects from policy waivers.

CSRF Protection for REST API when using Reverse Proxy Authentication

This release extends protection against cross-site request forgery (CSRF) to cover the case where the public REST API of IQ Server is exposed via reverse proxy authentication. As a consequence, clients accessing the REST API in combination with reverse proxy authentication need to be updated to include additional HTTP

headers in their requests to IQ Server when these requests can alter data. Please see Report-related REST APIs - v2 (see page 262) for details regarding the required changes.

Reduction of Ongoing Disk Space Growth

To reduce the storage requirements, especially during long-term use, this version of IQ Server no longer archives the binary fingerprints (`sonatype-work/clm-server/scan`) for all application evaluations. Going forward, only the binary fingerprints for the most recent evaluation of a given application and stage are kept on disk. The fingerprints used for a previous application evaluation at the same stage are automatically deleted.

Session Timeout Reset

To address security requirements, the IQ Server user interface now resets to the login state upon session expiration. This results in a loss of in-progress changes when your session times out. The session expires after thirty minutes of inactivity. To avoid loss of changes, complete all tasks prior to the timeout.

1.26 (March)

The IQ Server 1.26 release contains the following updates:

PKI Authentication for Integrations

Tools and plugins can be configured to use PKI authentication, which delegates authentication to the Java Virtual Machine (JVM). A reverse proxy server is required when using PKI authentication.

For more information, see Reverse Proxy Authentication in the IQ Server Setup (see page 96) topic.

Nexus IQ for Jenkins 2.x

A new Nexus IQ for Jenkins 2.x plugin is now available. For Pipeline projects, Nexus IQ for Jenkins 2.x allows for more advanced customization and automation for evaluating a Jenkins Workspace. Nexus IQ for Jenkins 2.x is best suited for new users running Jenkins 2.x.

For more information, see the Nexus IQ for Jenkins[10] topic.

Security Advisory

In this release, protection was added to reverse proxy authentication to address Cross-Site Request Forgery (CSRF) attacks at integration API endpoints. CSRF protection is now enabled specifically for requests authenticated via reverse proxy authentication, including SSO and PKI authentication.

---

10 https://help.sonatype.com/display/NXI/Nexus+IQ+for+Jenkins

All integrations in the 1.26 release support CSRF protection. Older integrations are not compatible and should be upgraded. If not upgraded, you may need to disable CSRF protection.

The public API is subject to CSRF attacks when made available through reverse proxy authentication. In typical use, the public API is for system-to-system integrations and in those scenarios reverse proxy authentication is not expected between the two systems. CSRF protection will be added to public APIs exposed via reverse proxy authentication in a future release.

For more information, please see Reverse Proxy Authentication (see page 268).

1.25 (January)

The IQ Server 1.25 release contains the following updates:

Support for Multiple LDAP Servers

IQ Server now supports authenticating against multiple LDAP servers, each with its own unique configuration. These updates allow you to add, remove, and reorder multiple LDAP servers. Once arranged, authentication searches the ordered list for a positive match on credentials.

For more information, see the LDAP Integration (see page 113) topic.

Support for Webhooks

Webhooks allow you to integrate IQ Server into your process. For example, you can set up webhooks to fire when application evaluations complete, letting you automate processes that tie into the IQ Server.

For more information, see the Webhooks (see page 278) topic.

Updates to Saved Filters

The name of a saved filter is now added to the file name when exporting data.

When you open a saved filter, the filter name now appears at the top of the Manage Filters menu.

For more information, see the Dashboard (see page 190) topic.

Custom JIRA Fields

JIRA Notifications can now be configured to supply a predefined value to any field via the Custom Fields section. This allows configuration of JIRA issue types which have required fields that are not configurable through the IQ Server UI.

For more information see JIRA Notifications in the Policy Management (see page 146) topic.

2016 Release Notes

⚠ Sonatype encourages using the most current IQ Server release and not trailing behind more than six months. Release notes for the most current versions can be viewed here (see page 13) .

1.24

The IQ Server 1.24 release contains the following updates:

IQ Server Authorization for Tools

The IQ Server provides extended functionality to a number of tools (e.g. Nexus Repository Manager, Nexus IQ for Hudson/Jenkins 1.x, Nexus IQ for Bamboo, IQ for Eclipse, CLM for Maven, etc.). Authentication and authorization was added in version 1.14 of IQ Server, but disabled by default for backwards compatibility.

Starting in version 1.24, this is now enabled by default; it forces tools to provide authentication details. While not recommended, if you need to preserve previous functionality, the IQ Server can be configured to allow anonymous access for all tools.

The affected tools include:

- Nexus IQ for Bamboo
- Nexus IQ CLI
- Nexus IQ for Hudson/Jenkins 1.x
- IQ for IDEA
- IQ for Eclipse
- CLM for Maven
- IQ for Nexus Repository Manager
- CLM for SonarQube

Lifecycle Container Analysis

You can now use the Nexus IQ CLI to evaluate Docker images in a Twistlock environment. Twistlock version 1.5.56 or newer is required for this integration.

For more information, see The Nexus IQ CLI chapter.

Export Dashboard Results

The Dashboard's View menu now has an Export Violations Data button that lets you download data displayed in the current view to a .CSV file for use in spreadsheets.

For more information, see the Dashboard[11] topic.

Saved Filters

The Filter menu on the left side of the Dashboard has a new Manage Filters menu. From here you can save Dashboard filter criteria allowing for recall and repeated use. Multiple saved filters can be created using different names.

For more information, see the Dashboard[12] topic.

Risk Severity Heatmap in Dashboard Results

The Dashboard Results now provide a heatmap representation of risk to highlight areas of concern. Results with lower risk are a light shade of blue and results with higher risk are a darker shade.

For more information, see the Dashboard[13] topic.

Updated IQ Server Plugins

The IntelliJ IDEA Plugin has been updated to support global configuration of the IQ Server.

Security Fixes

Fixed a cross site scripting (XSS) vulnerability exposed via display of usernames.

Notable Bug Fixes

Fixed a problem with Audit tab not showing some non-Java component changes.

1.23

The IQ Server 1.23 release contains the following updates:

11 https://help.sonatype.com/display/DWT/Dashboard
12 https://help.sonatype.com/display/DWT/Dashboard
13 https://help.sonatype.com/display/DWT/Dashboard

A New Look for the Dashboard

The Dashboard has a new look that reduces visual noise and makes it easier to see the overall health of components used in your development cycle.

New Dashboard Filter for Organizations

The Filter menu on the left side of the Dashboard has a new filter called Organizations. It allows you to view violations from one or more organizations, which includes violations from any applications attached to the organization(s). For more information, see the Dashboard[14] topic.

Component Intelligence for JavaScript Now Available

IQ Server now provides JavaScript component data for application evaluations just like it does for Java components (in addition to the existing npm JavaScript package support for repositories). Refer to the in-product announcement for known issues and updates.

Removed REST APIs v1

REST APIs v1 had the limitation of working only with Maven components while v2 works with any Java components, and (as of IQ Server 1.23) JavaScript components. Be sure to review the paths or URLs in your REST calls, and update them to v2 as needed. Any remaining v1 calls will return a “404 Not Found” message.

For more information about REST APIs v2, see the REST APIs[15] topic.

Updated Plugins

The following plugins were updated to resolve a vulnerability: Sonatype CLM for Maven, Nexus IQ for Bamboo, and Nexus IQ for Hudson/Jenkins 1.x.

Security Fixes

- Fixed a vulnerability with IQ Server causing resource leaks in a rare scenario that is unlikely to occur in most environments.
- Fixed an IQ Server local user account authentication vulnerability that uses a highly complex theoretical attack approach.
- Fixed an XML external entity (XXE) vulnerability affecting the Maven plugin and CI plugins.

14 https://help.sonatype.com/display/DWT/Dashboard
15 https://help.sonatype.com/display/DWT/REST+APIs

1.22

The IQ Server 1.22 release contains the following updates:

Enhanced Configuration for Proprietary Components

Proprietary components can now be defined at any level in the system hierarchy: Root organization, organization, or application. In addition, the configuration features for proprietary components have moved from the System Preferences menu to the Organization & Policy area.

If you use any of the following plug-ins, it's recommended that you install the latest version of the plug-in to take advantage of the changes to proprietary components configuration:

- Nexus IQ for Bamboo
- Nexus IQ for Hudson/Jenkins 1.x
- Sonatype CLM for Maven

Improved Dashboard Filters

On the Dashboard, it's now easier to see which filters are active and which are not. The following filter behaviors have been added:

When all items of a filter are selected, a total count is shown.
When only some items of a filter are selected, a counter (e.g. "2 of 4") is displayed.
When no filter items are available for selection, the filter is disabled.

Rebranded Roles and Permissions

The names of a few security roles and permissions have been updated as follows:

CLM Administrator is now Policy Administrator.
The Edit CLM Elements permission is now Edit IQ Elements.
The View CLM Elements permission is now View IQ Elements.

New Quick Start Guide

A Quick Start Guide for Nexus IQ Server has been added to the documentation. It'll help you get IQ Server up and running in minutes for the purpose of trying out its features before installing it in your development environment.

1.21

The IQ Server 1.21 release contains the following updates:

Nexus Firewall

The ability to send notifications when a new component violates policy in an audited repository has been added.

Integrations

- Added a new plugin for IntelliJ IDEA called IQ for IDEA. For details, see the IQ for IDEA chapter.
- Added a new integration for Atlassian JIRA with the ability to create JIRA tickets via IQ Server Notifications. For details, see Notifications in the Basic Policy Management chapter.
- Improved application analyzer (scanner) performance for all integrations.
- Added support for Atlassian Bamboo 5.10, 5.11, and 5.12.
- Rebranded the Bamboo plugin to Nexus IQ for Bamboo.
- Rebranded the Hudson/Jenkins plugin to Nexus IQ for Hudson/Jenkins 1.x. If you have a prior version of the plugin installed (called Sonatype CLM for Hudson and Jenkins), then you must uninstall the older version before installing the new rebranded one.

IQ Server UI

In the Policy Editor, Notifications and Actions have been split into separate sections. See the Basic Policy Management chapter for details.
In the Organization & Policy area, two commands have been added to the Actions menu:

Add ID to Clipboard - To copy the Application ID to the Clipboard for easy use when configuring an IQ Server plugin.
Change App ID - To change the Application ID for use with IQ Server plugins.

The Application ID now appears next to the Application Name near the top of the Organization & Policy area.

In the Dashboard area, the design of filters has been updated.

IQ Server Log Files

The configuration file (config.yml) for IQ Server shipped with this release contains updated defaults for the number of log files to archive (archivedFileCount). To ease troubleshooting, both the server and request logs are now being archived for the past 50 days. We recommend that customers with existing installations of IQ Server adopt this change for their configuration files after verifying that the disk holding the log files has sufficient free space left to account for the slight increase in storage requirements.

Security

Pre-Authentication Deserialization Vulnerability - Versions of IQ Server prior to version 1.18 are vulnerable to remote code execution via a deserialization bug. The affected library was upgraded in IQ Server 1.18, reducing the impact, but other exploits involving deserialization may exist.

IQ Server 1.21 implements a complete fix by removing access to the deserialization functionality. We recommend you upgrade to IQ Server 1.21 as soon as possible.

HTTP Header Vulnerability - All previous versions of IQ Server up to and including 1.20 are vulnerable to an attack due to HTTP Host header value injection. We recommend you upgrade to IQ Server 1.21 as soon as possible.

1.20

The IQ Server 1.20 release contains the following updates:

Nexus Firewall

The Nexus Firewall solution now has the ability to release a component from quarantine. Also, you can now override a vulnerability of a repository component. For more details, see the IQ for Nexus Repository Manager chapter.

New User Interface for Configuration

IQ Server configuration features have a new user interface designed to make it easier to configure policies, organize system hierarchy, and manage user access. In the Organization & Policies area, several items have been renamed:

- The Security tab is now the Access button.
- Tags are now called Application Categories.
- Labels are now called Component Labels.

Access Management

There are a few additions to managing user access:

- For security access, a new built-in group called Authenticated Users has been added. It contains any user who is authenticated by a realm integrated with IQ Server. You can assign Authenticated Users to roles in the same way as individual users or other groups.
- For user roles, there's a new permission called Add Applications. It allows you to grant users the ability to create applications within the scope of an organization without granting them full owner permissions. This new permission has been added to the CLM Administrator and Owner roles as well

as any custom role that grants permission to edit an organization. You may want to review custom roles and their assignments to verify permissions are set as desired.
- For the login process, feedback is now provided when a login fails due to LDAP issues.

System Hierarchy

You now have the ability to move an application from one organization to another. This makes it easier to reorganize the IQ Server system hierarchy when needed.

Component Data

For component data, there's a new license type called Not-Supported. It indicates that Sonatype or the target ecosystem does not currently support automated license collection for a component's format.

In addition, the Sonatype Data Research group is now providing the following data:

- Common Vulnerability Scoring (CVSS) for security issues that have reserved CVEs as of February 23, 2016.
- Root Cause information in the vulnerability details view.

Nexus IQ for Bamboo

The Nexus IQ for Bamboo plug-in has been updated to version 1.0.6. It fixes an error that can occur when connecting the Bamboo plug-in to IQ Server over SSL.

Performance

The loading times for the Dashboard, Reporting, and Organization & Policies areas have been significantly reduced, especially for systems with large data sets. If you have hundreds or thousands of organizations and applications, you should see faster loading of the Dashboard, reports, and configuration features. The Dashboard also has simplified summary results at the top, which makes it easier for you to focus on the Risk section.

1.19

The IQ Server 1.19 release contains the following updates:

Ability to Augment Repository Component Data

Several changes were made to the Firewall solution:

- In Repository Results, you can now waive violations, override licenses, and assign labels to repository components.

- You can delete repositories in IQ Server.
- Performance is improved when deleting Audit-enabled proxy repositories.

Enabled CLI for Auditor

Users of the Nexus Auditor solution can now access the Nexus IQ CLI features, which were previously available only for the Nexus Lifecycle solution. For information about Nexus IQ CLI, see the Nexus IQ CLI (see page 33) topic. For questions about your current solution or license, contact support@sonatype.com[16].

npm Data Available in Firewall

npm packages are now supported in Firewall. Available data includes: identification, licenses, and vulnerabilities.

New Logo for IQ Server

IQ Server has a new icon that appears on the IQ Server toolbar and its browser tab.

Updated IQ Plugins

The scope of analytics information collected by IQ Server for proprietary packages is narrower. This change is incorporated into all IQ plugins.

Additional Data for Sonatype Vulnerability Types

The IQ Server Application Composition Report and Repository Results include additional information for Sonatype vulnerability types.

Support for SonarQube 5.2 and 5.3

The SonarQube plugin version 1.0.4 now includes support for SonarQube 5.2 and 5.3.

16 mailto:support@sonatype.com

2015 Release Notes

⚠ Sonatype encourages using the most current IQ Server release and not trailing behind more than six months. Release notes for the most current versions can be viewed here (see page 13) .

1.18

The IQ Server 1.18 release contains the following updates:

New Root Organization

IQ Server now includes the Root Organization at the top of the system hierarchy, which acts as a container for all organizations in the system. Any policy set in the root organization is inherited globally by every organization and their associated applications. For more information about the Root Organization, see the Organization and Application Management chapter.

If you are upgrading from IQ Server version 1.17 or earlier, it is strongly recommended that you read the following documentation:

- The Root Organization[17] topic.
- Root Organization Best Practices[18], a Knowledge Base article.

New Audit and Quarantine Features

For users of the Firewall solution, you can now evaluate repositories in your Nexus Repository Manager. When a repository is evaluated, use the Audit feature to identify policy violations associated with components inside the proxy repository. Additionally, you can use the Quarantine feature to prevent newly proxied components with policy violations from being served to clients. The evaluation results are provided in the new Repository Results (similar to the Application Composition Report). Note: At this time, only Maven and NuGet packages inside proxy repositories will provide results. For more information, see IQ Server and Repository Management[19].

Documentation Updates

Rebranding

17 https://help.sonatype.com/display/DWT/Root+Organization
18 https://support.sonatype.com/hc/en-us/articles/214078978-Root-Organization-Best-Practices
19 https://help.sonatype.com/display/DWT/IQ+Server+and+Repository+Management

- Screenshots and text were updated to reflect the new name, IQ Server. While there are a few chapters about plug-ins that need to be rebranded, the majority of the documentation is finished.

Licensing

- Information about the products and licenses required to use the features discussed in a chapter has been added to the beginning of each chapter. The purpose is to help clarify the role of IQ Server in the licensed Nexus solutions: Firewall, Auditor, Lifecycle and Repository. To learn more about Nexus solutions and licenses, see IQ Server (see page 41).

1.17 (our new name)

The IQ Server 1.17 release contains the following updates:

Rebranding

The Sonatype CLM rebranding and renaming effort is at the heart of this release. Going forward, Sonatype CLM is no more (oh how we miss thee). While you may still see this in some areas/screenshots, we believe this change will help add clarity to the symbiotic relationship between our products (IQ Server and Nexus Repository Manager). We also realize this may cause some confusion, and appreciate your patience as we move forward. In the event you have any scripts that use the previous binary names you will want to update those as part of your upgrade process.

Sonatype CLM for Maven and CI

As part of this release, a bug was fixed in Sonatype CLM for Maven. The bug fix addressed an issue where a POM file could be included among the identified files. This has been addressed so that the dependencies will still be used and identified, however, the POM file won't be included. We recommend all users update to the latest version of Sonatype CLM for Maven. For anyone using Sonatype CLM for CI (Hudson/Jenkins or Bamboo) in combination with Sonatype CLM for Maven, we recommend you upgrade both integrations.

1.16

The Sonatype CLM 1.16 release contains the following updates:

This release represents a variety of improvements that came directly from customer requests. From adding improved API support and a new UI for viewing organizations and applications, there's a little something for everyone. We even took the opportunity to update the way certain policy condition situations are handled. The summary of improvements is listed below, followed by details and links to the updated/new documentation.

Improved Policy Conditions for Security Vulnerabilities

To resolve situations where policy violations for components with more than one security vulnerability were produced, the evaluation for conditions about security vulnerabilities has been revised. Going forward, in situations where a constraint has multiple conditions, and all must be satisfied to trigger the constraint, if there are multiple vulnerabilities for a component, at least one vulnerability must meet all vulnerability-related conditions.

As a result of this change, some of the policy violations related to vulnerabilities that you observed in the past will resolve themselves after reevaluation with the new version of Sonatype CLM.

To better illustrate the expected changes, consider a component with two vulnerabilities, one with severity 9 and status *Not Applicable* and one with no severity and status *Open*. When evaluated against the Sonatype sample policies, prior versions of CLM reported violations of the *Security-Unscored*, *Security-Low*, *Security-Medium* and *Security-High* policies. After the update, only the *Security-Unscored* policy will be violated.

New API for Component Details

This new API allows you to gather information about any component known to Sonatype CLM. The details provided won't list policy violation information (that's already covered with the Component Details by Report API).

New CLM Server Option for CSRF Protection

To prevent unwanted attacks via cross-site request forgery, a new configuration item was added.

New UI for Viewing Organizations and Applications

In the Managing Organizations and Applications area of Sonatype CLM Server, the navigational list has been changed to a tree view in order to improve the display of organizations and applications and their parent-child relationships. A filter box has been added to allow for easy searching and filtering of organizations and applications by name.

Support for Reverse Proxy Authentication for Single Sign-On (SSO)

The Sonatype CLM Server now supports additional configuration options for reverse proxy authentication for single sign-on (SSO).

Reduced LDAP Connection Retry Delay

To reduce the delay in reconnecting to an LDAP Server when a connection is lost, the default value of the Retry Delay setting has changed from 300 seconds (5 minutes) to 30 seconds. If you want a different

reconnection interval, you can manually change this setting in the LDAP Administration area of Sonatype CLM Server.

1.15

The Sonatype CLM 1.15 release focuses predominantly on improving security administration functionality. As part of this, you will likely notice some changes with regard to the associated interface in these areas.

Built-in Roles

Previously, the built-in (i.e. default) roles were only visible at the Organization and Application level, or in the *Global Roles* menu area. However, all available built-in roles can now be viewed from the System Preferences area.

Using the *Roles* menu option, you can look into exactly what permission(s) each role has.

Calculate Trends

The Violation Summary located on the CLM Server dashboard is calculated for all apps over all time. In installations with a large number of applications, this can take a fair amount of time. Based on user feedback around the general use of this feature, as well as the situation where this can make the dashboard feel unresponsive, we have moved this to a manual calculation.

Custom Roles

If the built-in roles don't provide the options you are looking for, create custom roles. This latest feature provides the ability to name and describe a completely new role, as well as define which permissions each role has.

Java 1.8 Compatibility

The Sonatype CLM Server now supports the Java 1.8 runtime.

⚠ This is in addition to a previous update that allows Sonatype CLM to analyze Java8 bytecode.

1.14.2

The Sonatype CLM 1.14.2 release contains the following updates:

Security

- Cross site scripting (XSS) vulnerability
- LDAP injection vulnerability

Performance

- Dashboard performance improvements
- Component details (via Dashboard) performance improvements

1.14

The latest version of Sonatype CLM is free for all existing users of Sonatype CLM. This includes the Sonatype CLM Server as well as the entire Sonatype CLM suite of tools (e.g. Sonatype CLM for Nexus).

The following updates are included in the 1.14 release:

Notification Panel

A new notification panel located next to the name of the logged-in user provides a mechanism for the Sonatype CLM development team to communicate directly with Sonatype CLM users. Look to this location for important announcements that affect your CLM Server.

Sonatype CLM Authorization

The Sonatype CLM Server provides extended functionality to a number of tools (e.g. Sonatype CLM for Nexus, Hudson, Jenkins, Bamboo, Eclipse, Maven, etc.). Previously these tools allowed limited, or no direct, authorization options when evaluating applications.

Starting with Sonatype CLM 1.14, CLM Server authorization for these tools is optional by default. This means a username and password can be entered if desired. Additionally, the Sonatype CLM Server can be configured to force authorization for all tools.

If you desire to turn off the anonymous access, we recommend you upgrade your Sonatype CLM Server first, and then follow with the various tools. In cases where you can't upgrade the tools as quickly or easily as the Sonatype CLM Server, we recommend waiting until those tools are updated before forcing authorization.

The affected tools includes Sonatype CLM for:

- Bamboo
- CLI
- Hudson/Jenkins
- IDE (Eclipse)

- Maven
- Nexus
- SonarQube

Role-based Notifications and Monitoring

The Sonatype CLM Server allows notifications and monitoring to be configured such that when a policy violation occurs, users will be notified. Previously, policy notifications and monitoring required an email to be added.

In the Sonatype CLM 1.14 update, users can select a particular role in addition to entering a specific email. When policy violations occur, any user assigned to that role will be emailed.

CLM Server Config Update

An update to the application log has been made. These changes provide a foundation for more detailed logging in the future. Previous users of Sonatype CLM who are upgrading to 1.14, and want to take advantage of this feature, will need to update their logFormat configuration.

Please review the `config.yml` file included with the Sonatype CLM Server download. An example of the new logging is provided below:

```
2015-04-10 10:34:16,919-0400 INFO [qtp308511037-32 - GET /rest/productNotifications?
timestamp=1428676456892] admin com.sonatype.insight.brain.notifications.HdsProductNotificationService -
Updating notification cache from HDS
```

Vulnerability Details in Application Report

The Edit Security Vulnerability area of the Component Information Panel (CIP) located in the Application Composition Report has been modified. A new information column has been added with an icon in each row. Clicking on this icon will display a summary of the Security Vulnerability Information Sonatype has curated.

1.13

The 1.13 release of Sonatype CLM encompasses a number of the related CLM tools including:

A Special Note About the CIP

This release features a number of improvements that affect the Component Information Panel (CIP). The CIP is the graphical display used in a number of CLM tools including Nexus, Eclipse, and the Application Composition Report.

Nexus CompatibilityFor Nexus users, if you upgrade to version 1.13 of the Sonatype CLM Server, you must also upgrade Nexus to version 2.11 or higher. Failure to do so will result in errors in the Nexus CIP due to an incompatibility between CLM and Nexus.Report Viewing and Browser CachingOur recommendation is that the browser's cache be cleared once the CLM Server has been upgraded to 1.13. This will prevent issues when interacting with previously opened reports.

Without clearing the browser's cache users may experience undesirable behavior such as HTTP errors, or appear like the report functionality is broken.

⚠ This has been corrected in 1.13, and by clearing the browser's cache as part of this upgrade, it will not be required in future versions.

What's New in Sonatype CLM 1.13?

Sonatype Data Services

A number of improvements around providing data will be reflected in this release, and will improve the quality of data Sonatype provides. The most notable impact seen in this release includes the update of license related information.

Because of these data related changes, Sonatype recommends all users delete their browsing cache to ensure the latest and most up-to-date information is provided.

REST/API Functionality

A number of new publicly available APIs are now available including API functionality allowing you to:

- Evaluate a component against specific policy.
- Retrieve policy violation information.
- Retrieve report information.

Sonatype CLM Server (Web Application)

As a result of Sonatype Data Services updates, a number of user interface improvements have been made. This includes:

- Dots, dashes, and underscores are now permissible in CLM related names (e.g. Application, Policy, etc.).
- License types available to License Threat Groups.
- Policies that detect unassigned licenses can now be created.

CLM for Hudson and Jenkins

For users of Sonatype CLM for Hudson and Jenkins, users now have the ability to:

Specify CLM stage

Configure if a build fails due to a CLM failure.

This is provided in the latest CLM for Hudson and Jenkins documentation.

CLM for Maven

Scope is now configurable in CLM for Maven. This functionality is described in the latest update of the CLM for Maven documentation.CLM for EclipseSimilar to the CLM for Maven update, CLM for Eclipse now allows scope to be configured. This functionality is described in the latest update of the CLM for Eclipse documentation.

Help and Documentation

While all features and updates described in these release notes are included in the latest documentation, work has been made to improve access to documentation specific to the version of CLM.

Specifically, from this release forward, when clicking on the Help menu item within the CLM Server Web Application, a user will be taken to the matching version of the documentation. Once there, any search of the documentation will only return those results related to that specific version.

Additional Updates

- Removed Bulk license and SV editor tools from Application Composition Report.
- Increased max allowed components for CLM scans.
- Allow manual selection of multiple license options for a component via the Application Composition Report.
- Support for Operate phase in the CLM Command Line Scanner (CLI).
- Various Other Bug Fixes.

2014 Release Notes

⚠ Sonatype encourages using the most current IQ Server release and not trailing behind more than six months. Release notes for the most current versions can be viewed here (see page 13) .

1.12

The team has been listening to your feedback, and working to improve how you interact with the Sonatype CLM Server, and the 1.12 release reflects that. We've tweaked and polished, organized and decluttered, added color and changed fonts.

While the UI improvements are the most noticeable, don't let those distract you from a number of additional enhancements to the Sonatype CLM Server as well. Here are the areas that have had improvements in this release:

- Sonatype CLM Dashboard
    - Filter
    - Policy Violations Summary
    - Navigation
    - Overall Performance

- New Policy Violations API
- Application Composition Report
    - License Analysis
    - Security Vulnerability Scoring

- Various Bugs

Affected CLM Tools

The majority of features in this update focus on the Sonatype CLM Server, and will require an upgrade. If you are using any of the following components, you should be sure to upgrade them as well.

- Sonatype CLM CI Plugin
- Sonatype CLM IDE Plugin (Eclipse)
- Sonatype Stand-alone (Command Line) CLM Scanner

What's New in Sonatype CLM 1.12

Sonatype CLM Dashboard

Outside of the changes to colors and fonts, which improve readability and use, several areas of the dashboard have also been enhanced.

Updated Filter

The filter has been been moved into an expandable and collapsible drawer on the left side of the Dashboard. The filter will also now display which filters are in use, and how many selections have been made. You can read more about using the filter in the Filters section of the Dashboard User Guide.

Policy Violations Summary

A new category, Waivers, has been added. In addition, average age value and the 90th percentile value for age have been added to indicate how long a component has been in a particular category.

Navigation and Overall Performance

The breadcrumb navigation for the Dashboard has been improved such that when clicking on the Dashboard breadcrumb will return you to the correct tab within the Dashboard.

In addition to the above, efforts have been made to improve the overall performance of the Dashboard.

New Policy Violations API

We've updated the Sonatype CLM REST APIs to include the ability to retrieve Policy Violation information.

Application Composition Report

Two enhancements have been made to the way License and Security information are displayed. The details have been provided below.

License Analysis

The License Analysis area has been updated so that effective licenses are now displayed.

Security Vulnerability

Previously, security vulnerabilities with a level 7 CVSS score were included in the Severe category and indicated with the color orange. These have been moved into the Critical category, which is indicated with the color red. This brings this type of vulnerability into better alignment with the NVD scoring system.

Licensing and Features

Licensing

Nexus solutions — Nexus Lifecycle, Nexus Auditor, and Nexus Firewall — are powered by Nexus IQ (Server). Each solution has a corresponding license that controls the availability of IQ Server tools and features. You may have access to a portion or all of IQ Server tools and features depending on the solution you purchased.

In general, when we are talking about a solution, we are referring a particular license (e.g. Nexus Lifecycle vs. Nexus Firewall) and the features it unlocks. In contrast, when we describe a product, we specifically mean the thing you will install (e.g. Nexus IQ Server). The various solutions (licenses) and their corresponding product are broken down in the table below:

Solution (License)	Product (What's Installed)
Nexus Auditor	Nexus IQ (Server)
Nexus Firewall*	Nexus IQ (Server) + Nexus Repository (Manager)
Nexus Lifecycle	Nexus IQ (Server)

Feature Matrix

As you may have noticed, in some cases the same product is installed regardless of the license. Use the following matrix to help understand which features are unlocked by a particular license:

Feature	Nexus Auditor	Nexus Firewall	Nexus Lifecycle
Organization Management	👍	👍	👍
Application Management	👍	🚫	👍
Policy Management	👍	👍	👍
Continuous Monitoring	👍	🚫	👍
Stages			
• Proxy	🚫	👍	🚫

Feature	Nexus Auditor	Nexus Firewall	Nexus Lifecycle
• Develop	🚫	🚫	👍
• Build	🚫	🚫	👍
• Stage	🚫	🚫	👍
• Release	👍	🚫	👍
• Operate	🚫	🚫	👍
Integrations			
• Repository	🚫	👍	🚫
• CLI	👍	🚫	👍
• IDE	🚫	🚫	👍
• Build and Continuous Integration	🚫	🚫	👍
• SonarQube	🚫	🚫	👍
Dashboard and Reporting			
• Dashboard	👍	🚫	👍
• Application Composition Report	👍	🚫	👍
• Repository Results	🚫	👍	🚫
Platform Capabilities			
• REST API	🚫	🚫	👍
• WebHooks	🚫	🚫	👍

> *The Nexus Firewall solution requires installation of both Nexus IQ (Server) and Nexus Repository (Manager). For more information, please see IQ Server and Repository Management[20] .*

Download and Compatibility

You should have already received an email containing the necessary information to unlock all the Nexus IQ Server features you have purchased. If you have not received this, please contact your Sales executive directly, or send an email to support@sonatype.com[21].

Depending on your purchase, you may need to install additional tools such as the Bamboo or Jenkins plugin. No matter what was purchased, you will need to download and install Nexus IQ Server first.

If you would like to see a description of the latest features, as well as those for a specific release, view our Release Notes (see page 13).

⚠ Starting with the 1.17 release, *Sonatype CLM* was renamed to *Nexus IQ Server*. You may still occasionally see Sonatype CLM in the product or documentation.

20 https://help.sonatype.com/display/NXIM/IQ+Server+and+Repository+Management
21 mailto:support@sonatype.com

Latest Releases

Sonatype Application	Latest Version	Download
IQ Server & CLI	1.49.0	Starting in 1.45, Nexus IQ Server uses a more effective and condensed format to store policy violation data. Depending on the database size of your installation and the hardware you use, upgrading your database to this new format can take notable time (up to hours in the worst case scenarios). You should review the upgrade procedure detailed in Upgrading the IQ Server to Version 1.45 (see page 93) . nexus-iq-server-latest.tar.gz[22] (ASC[23], MD5[24], SHA1[25]) nexus-iq-server-latest.zip[26] (ASC[27], MD5[28], SHA1[29]) This bundle already contains the CLI tool for convenience.
Repository Manager 2	2.x	Please see Download Repository Manager 2[30]
Repository Manager 3	3.x	Please see Download Repository Manager 3[31]

22 https://download.sonatype.com/clm/server/latest.tar.gz
23 https://download.sonatype.com/clm/server/latest.tar.gz.asc
24 https://download.sonatype.com/clm/server/latest.tar.gz.md5
25 https://download.sonatype.com/clm/server/latest.tar.gz.sha1
26 https://download.sonatype.com/clm/server/latest.zip
27 https://download.sonatype.com/clm/server/latest.zip.asc
28 https://download.sonatype.com/clm/server/latest.zip.md5
29 https://download.sonatype.com/clm/server/latest.zip.sha1
30 https://help.sonatype.com/display/NXRM2/Download
31 https://help.sonatype.com/display/NXRM3/Download

Bamboo	1.8.1	nexus-iq-bamboo-plugin-1.8.1-03.obr[32] (ASC[33] , MD5[34] , SHA1[35])
Jenkins	3.x	Available to update through the Jenkins Update Center[36]
Hudson/ Jenkins 1	2.21.0	⚠ This plugin is deprecated and is no longer receiving updates. We suggest that you uninstall this plugin and then install the Jenkins 3.x plugin; however, there is no automated process to migrate jobs between plugins. nexus-iq-jenkins-plugin-2.21.0-01.hpi[37] (ASC[38], MD5[39], SHA1[40])
Eclipse	2.13.0	Eclipse Update Site: https://download.sonatype.com/clm/eclipse/releases ⚠ The link above is not directly browsable, enter it into the list of update sites in Eclipse under "Help - Preferences - Install/Update - Available Software Update Sites. If you need an offline copy of this site, see here[41] for mirroring instructions. IBM Rational Application Developer (RAD) is not tested, but should work. Note that the plugin requires Eclipse 3.6 and greater, RAD versions based on Eclipse older than this will not work.

32 https://download.sonatype.com/clm/ci/bamboo/nexus-iq-bamboo-plugin-1.8.1-03.obr
33 https://download.sonatype.com/clm/ci/bamboo/nexus-iq-bamboo-plugin-1.8.1-03.obr.asc
34 https://download.sonatype.com/clm/ci/bamboo/nexus-iq-bamboo-plugin-1.8.1-03.obr.md5
35 https://download.sonatype.com/clm/ci/bamboo/nexus-iq-bamboo-plugin-1.8.1-03.obr.sha1
36 https://plugins.jenkins.io/nexus-jenkins-plugin
37 https://download.sonatype.com/clm/ci/nexus-iq-jenkins-plugin-2.21.0-01.hpi
38 https://download.sonatype.com/clm/ci/nexus-iq-jenkins-plugin-2.21.0-01.hpi.asc
39 https://download.sonatype.com/clm/ci/nexus-iq-jenkins-plugin-2.21.0-01.hpi.md5
40 https://download.sonatype.com/clm/ci/nexus-iq-jenkins-plugin-2.21.0-01.hpi.sha1
41 https://support.sonatype.com/hc/en-us/articles/224925028

Sonar Qube	1.2.0	sonatype-clm-sonarqube-1.2.0-02.jar[42] (ASC[43], MD5[44], SHA1[45]) ⚠ This plugin is only compatible with the 6.x branch of SonarQube and not the 7.x branch where the dashboard has been removed.
CLI	1.49.0	nexus-iq-cli-1.49.0-01.jar[46] (ASC[47], MD5[48], SHA1[49])
Maven	2.8.1	This plugin is automatically downloaded when invoked.
IDEA	2.0.0	nexus-iq-idea-plugin-2.0.0-01.zip[50] (ASC[51], MD5[52], SHA1[53])
Visual Studio	1.0	The "Nexus IQ for Visual Studio" extension is installable from within Visual Studio using the Extensions and Updates dialog box. The extension may also be downloaded from the Visual Studio Marketplace[54].

Clouds and Containers

Sonatype Nexus Repository Manager and IQ Server are offered via various cloud services and container formats. More information is available in the Integration space under Cloud Deployments[55].

IQ Server to Integration Plugin Version Compatibility

Sonatype encourages using the most current IQ Server release and not trailing behind more than six months. If you are using an older version of Nexus IQ Server, or any of the associated integrations, and wish to upgrade to the latest version, please contact our Support Team[56].

---

42 https://download.sonatype.com/clm/sonarqube/sonatype-clm-sonarqube-1.2.0-02.jar
43 https://download.sonatype.com/clm/sonarqube/sonatype-clm-sonarqube-1.2.0-02.jar.asc
44 https://download.sonatype.com/clm/sonarqube/sonatype-clm-sonarqube-1.2.0-02.jar.md5
45 https://download.sonatype.com/clm/sonarqube/sonatype-clm-sonarqube-1.2.0-02.jar.sha1
46 https://download.sonatype.com/clm/scanner/nexus-iq-cli-1.49.0-01.jar
47 https://download.sonatype.com/clm/scanner/nexus-iq-cli-1.49.0-01.jar.asc
48 https://download.sonatype.com/clm/scanner/nexus-iq-cli-1.49.0-01.jar.md5
49 https://download.sonatype.com/clm/scanner/nexus-iq-cli-1.49.0-01.jar.sha1
50 https://download.sonatype.com/iq/idea/nexus-iq-idea-plugin-2.0.0-01.zip
51 https://download.sonatype.com/iq/idea/nexus-iq-idea-plugin-2.0.0-01.zip.asc
52 https://download.sonatype.com/iq/idea/nexus-iq-idea-plugin-2.0.0-01.zip.md5
53 https://download.sonatype.com/iq/idea/nexus-iq-idea-plugin-2.0.0-01.zip.sha1
54 https://marketplace.visualstudio.com/items?itemName=SonatypeIntegrations.NexusIQforVisualStudio
55 https://help.sonatype.com/display/NXI/Cloud+Deployments
56 mailto:support@sonatype.com

⚠ Some versions of Nexus IQ Server (1.26 and above) are not fully compatible with old integrations. In cases where reverse proxy authentication is used, communication failures could occur between old integrations and the IQ Server. Integrations should be updated to the latest version when using reverse proxy authentication. For more information, please see our documentation on Upgrading the IQ Server (see page 90).

Compatibility of IQ Server with versions of our associated integrations are:

IQ compatibility with Nexus Repository Manager 2

IQ Server Version	Repository Manager 2 Version	Compatibility
All versions	2.12 and higher	✔

IQ compatibility with Nexus Repository Manager 3

IQ Server Version	Repository Manager 3 Version	Compatibility
1.35 and higher	3.6 and higher*	✔
1.34 and lower	3.6 and higher*	⊘
1.33 and higher	3.5**	✔
1.32 and lower	3.5**	⊘

> **Nexus Repository Manager 3.8 is the first version that fully supports Firewall with High Availability Clustering (HA-C).*
> ***Nexus Repository Manager 3.5 is the first version that has Firewall properly supported for all repository formats (including NuGet). Previous versions have known issues and are not listed on this page.*

IQ compatibility with Bamboo Plugin

IQ Server Version	Bamboo Plugin Version	Compatibility
All versions	1.1 through 1.7	✔

1.45 and higher	1.8	✓
1.44 and lower	1.8	⊘

IQ compatibility with Jenkins Plugin

IQ Server Version	Jenkins Plugin Version	Compatibility
1.45 and higher	3.1.x and higher	✓
1.26 through 1.44	1.1 through 3.0.x	✓
1.25 and lower		⊘

IQ compatibility with Hudson/Jenkins 1 Plugin

⚠ This plugin is deprecated and is no longer receiving updates. We suggest that you uninstall this plugin and then install the Jenkins 3.x plugin; however, there is no automated process to migrate jobs between plugins.

IQ Server Version	Hudson/Jenkins 1 Plugin Version	Compatibility
All versions	2.13 through 2.21	✓

IQ compatibility with Eclipse Plugin

IQ Server Version	Eclipse Plugin Version	Compatibility
All versions	2.8 through 2.13	✓

IQ compatibility with Maven Plugin

IQ Server Version	Maven Plugin Version	Compatibility
All versions	2.3 through 2.8	✓
1.45 and higher	2.8.1	✓
1.44 and lower	2.8.1	⊘

IQ compatibility with SonarQube Plugin

IQ Server Version	SonarQube Plugin Version	Compatibility
All versions	1.0 through 1.2	✓

IQ compatibility with IDEA Plugin

IQ Server Version	IQ for IDEA Plugin Version	IDEA Version	Compatibility
All versions	2.x and higher	15.x through 2018.x	✓
All versions	1.x	14.x through 15.x	✓

IQ compatibility with Visual Studio Plugin

IQ Server Version	Visual Studio Plugin Version	Compatibility
1.32 and higher	1.0	✓
1.31 and lower	1.0	⊘

System Requirements

Overview

The IQ Server is typically deployed on dedicated hardware. More specific hardware requirements are ultimately a function of the deployment architecture, the primary usage patterns and the scale of deployment.

With these influencing factors in mind, a general recommendation is provided as a starting point.

Development, test, or evaluation deployments can be scaled smaller than the recommendations and will continue to function, while performance degradation may be observed.

Installation Requirements

Item	Description
CPU and RAM	Recommend a processor with at least 8 CPU cores and 8GB of RAM for initial setup. A minimum of 6GB of process space should be available to the IQ Server. Additional RAM can improve the performance due to decreased disk caching. Example: Dual Intel Xeon E5620 with 2.4Ghz, 12M Cache, 5.86 GT/s QPI, Turbo, HT.
Disk	Storage requirements range with the number of applications projected to use the IQ Server. 500 GB to 1 TB of free disk space should provide more than adequate resources. Monitoring disk-space usage will help you gauge the storage needs in your actual deployment and react to growing demands in time. The IQ Server is an I/O intensive application and disk speed will affect the performance of the IQ Server considerably. We therefore recommend to use local drives or SAN usage. Usage of network mapped storage via NFS or similar is not recommended. It is important to consider the I/O load when running IQ Server in a virtual machine, especially when other virtual machines on the same host are running other I/O intensive applications e.g. the Nexus Repository Manager.

Operating System	Generally, any machine that can run a supported Sun/Oracle Java version should work. Refer to the Oracle documentation for specifics: Oracle JDK 8 and JRE 8 Certified System Configurations. The most widely used operating system for the IQ Server is Linux and therefore customers should consider it the best tested platform.
User Account	It is recommended that an unprivileged service account be created if running the IQ Server as a daemon.

Ports	The IQ Server requires the following network access.
	Inbound:
	• 8070 TCP: Used by all IQ Server clients for HTTP access. This port is configurable.
	• 8071 TCP: Used by the local host or other IT monitoring tools for monitoring and operating functions.
	  • This port is optional and configurable. If not specified, port 8081 will be used.
	Outbound:
	• 443 TCP to https://clm.sonatype.com : Used by the IQ Server to securely access Sonatype Data Services. This hostname and port are not configurable.
	  ⚠ Sonatype Data Services must be reachable by IQ Server on the following URL: https://clm.sonatype.com/ .

Java	Oracle Java 8 64 bit

Browser Requirements

Browser	Version
Internet Explorer	• IE 9 • IE 10 • IE 11
Firefox	• ESR (extended support release) • “Stable”

Chrome	• "Stable"
Safari (on OSX)	• 5.1.9 corresponding to OS X 10.6 • 6.0.4 corresponding to OS X 10.7 and 10.8 • 7.1 corresponding to OS 10.9

⚠ JavaScript must be enabled for the chosen browser.

REST API Requirements

The IQ Server REST APIs are versioned. As a best practice we recommend using the latest version of the IQ Server in addition to the latest version of the REST APIs. This ensures your system will take advantage of the latest features and improvements.

However, we also realize that users of previous versions need to maintain this compatibility even when there is an update. For this reason, we do provide support for previous versions based on the criteria below.

Supported API Versions:

- Sonatype CLM 1.12 and earlier - *Only* Supports REST API v1
- Sonatype CLM 1.13 and later - Supports *both* REST API v1 and v2
- IQ Server 1.17 to 1.22 - *Recommend* usage of v2 REST API only.
- IQ Server 1.23 and later - *Only* Supports REST API v2.

Identifying the version of the API is simple. Below we have provided an example using the REST API for retrieval of an organization ID:

```
http://localhost:8070/api/v2/organizations
```

As you can see, the **`v2`** located just after **`api`** indicates the version of the API. If you find that the API version you are using is not documented, and would like information on upgrading to the latest version you can contact our support team for assistance.

Sonatype Vulnerability Data

What is Sonatype Vulnerability Data?

Sonatype creates its data using a proprietary, automated vulnerability detection system that monitors, aggregates, correlates, and incorporates machine learning from publicly available information. We gather

data from various sources including the National Vulnerability Database, website security advisories, email lists, GitHub events from all open source projects, blogs, OWASP, OSS Index, Twitter, and customer reports. We have evaluated many paid-for services and have found the quality and precision of the data to be of limited value, driving our decision to build an intelligent, automated vulnerability detection system. The Sonatype Data Research team is not in the business of simply aggregating public security related feeds — we create the precise data we use.

How does Sonatype provide high quality data?

There are two considerations for data quality: (1) content of the security advisory and (2) precision of associating the content to the correct artifact. Automated decisions require extremely precise artifact identification and corresponding association of security information. Without accurate identification and association there is a high degree of false positives. We recently conducted a study of 6000 of the most popular Java components and found that name-based security association algorithms used by every tool other than Sonatype resulted in:

- 4500 correct non-issue identifications
- 1034 true positives
- 5330 false positives when the advisory identified CPE was part of the component name
- 2969 false negatives when the advisory identified CPE was not in the component name

False positives incur unnecessary research and upgrade costs. False negatives leave you at risk because there are no indicators that show you may be at risk. Sonatype uses a combination of automated identification and human research that eliminates false positives and negatives. This results in savings in research time to prove false positives and rework time to upgrade when not required.

What Data Does Sonatype Provide?

- The source of the advisory: Sonatype Security Research or the National Vulnerability Database
- The severity of the issue: CVSS and scoring system version and the source of the score creation
- The Common Weakness Enumeration (CWE)
- The exact description from the advisory
- A detailed explanation of the advisory risk and the attack vector (because the advisory description is often very poor)
- How to determine if you are vulnerable
- A recommendation on how to fix or work-around the issue
- The root cause of the issue; the exact class and vulnerable version range that was found in your code
- Publicly known attack vectors or exploits; additional resources that describe the exact issue

How is a vulnerability score / severity calculated?

Sonatype uses the Common Vulnerability Scoring System[57] (CVSS) to score vulnerabilities.

If a vulnerability identifier is prefixed with *SONATYPE*, then the vulnerability severity is its CVSS version 3 score.

If a vulnerability identifier is prefixed with *CVE*, then the vulnerability severity is its CVSS version 2 score.

Sonatype plans to ultimately migrate all CVSS version 2 scores to version 3. If a version 3 score is not available, the score will remain version 2.

Where are the Source Components?

Component binaries come from popular repositories like Central[58], NuGet.org[59], npmjs.org[60], Fedora EPEL, and PyPI[61]. We will also ingest components directly from GitHub, and other project download sites, when nominated by customers.

Binary repositories provide the ability to extract information like declared licenses, popularity, and release history. Additional component metadata comes from a variety of sources including direct research.

When is Vulnerability Data Available?

Sonatype Data Services are continuously updated, allowing the most recent data to be visible the instant a Nexus Lifecycle analysis occurs. This is true for both newly published components and newly discovered security issues. We have two processing queues for security vulnerabilities to ensure immediate availability of security data to our customers:

Fast Track - Our automated vulnerability detection systems process the various data sources each day. Upon issue discovery, the issue is validated by a researcher to ensure the correct component was identified, a brief issue description exists, and the vulnerable version range matches the advisory. This process generally makes newly discovered vulnerabilities available in 1-3 days depending on severity of the issue.

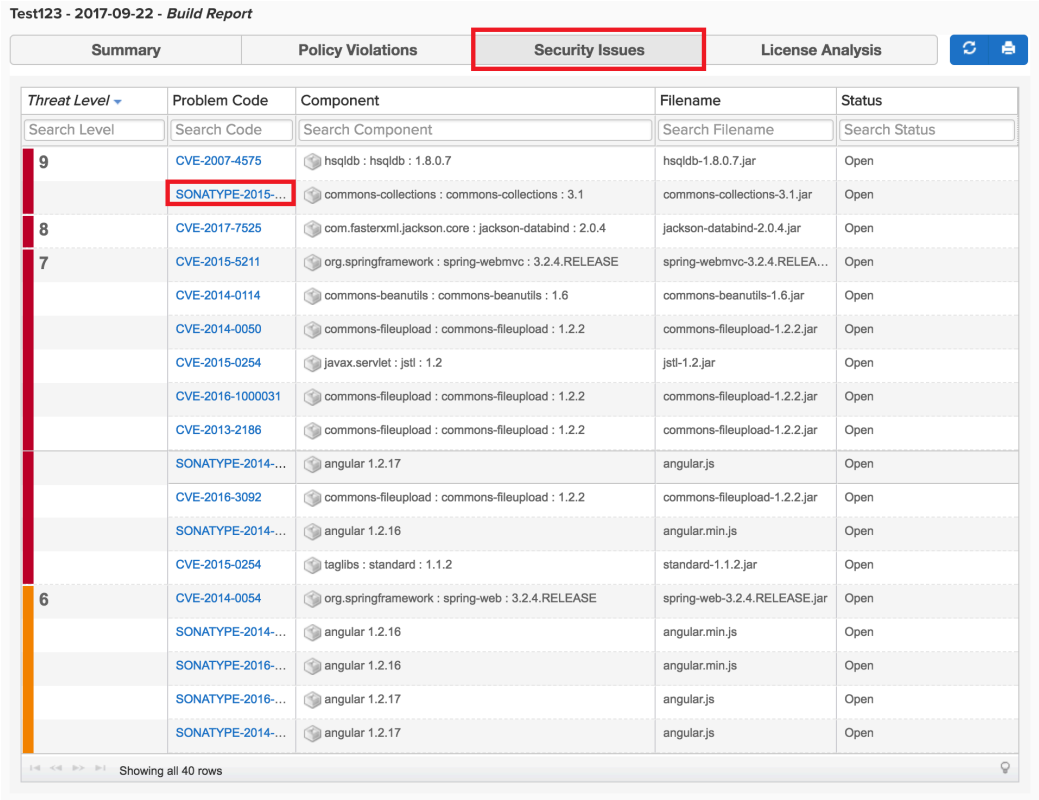
Deep Dive - The deep dive queue is a more methodical approach to ensure the issue has a clear explanation and fix recommendation. This is also where the source code of each issue is investigated to ensure the vulnerable version range is accurate. This process generally takes 5+ days, but can take less for issues deemed critical.

---

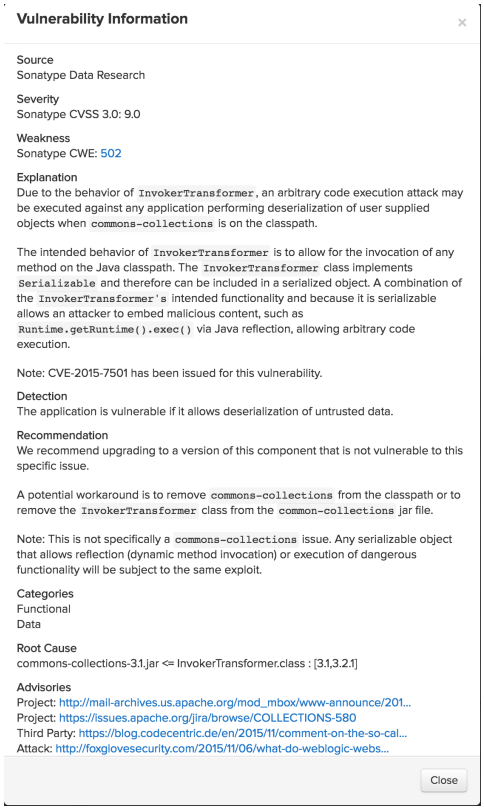
57 https://en.wikipedia.org/wiki/Common_Vulnerability_Scoring_System
58 http://search.maven.org/
59 http://nuget.org/
60 http://npmjs.org/
61 https://pypi.python.org/pypi

How do I Access Vulnerability Information?

Sonatype-enriched vulnerability data is available from the IQ Server Application Composition Report (see page 193). Select the *Security Issues* tab and then select the problem code you're investigating:



Then view the detailed Vulnerability Information:



You can also access this information from the *Vulnerabilities* tab of the Component Information Panel (see page 202).

How do I Use Vulnerability Information?

The important thing to remember is that evaluating your application and seeing security vulnerabilities should create motivation for further investigation.

For example, if it's recommended to do a component upgrade, use the CIP to identify a recommended non-vulnerable, popular version.

If the new version has the same API as the previous component, simply run unit and integration tests and make sure everything passes to successfully remediate the policy violation.

Nexus Lifecycle Quick Start

This guide can help you get IQ Server up and running for the purpose of trying out the associated Nexus Lifecycle functionality before installing it in your development environment. It should take approximately 15 minutes to complete using reference policies and applications.

> Nexus Lifecycle requires a license in order to experience the functionality described in this guide. If you are looking to try or purchase Nexus Lifecycle, schedule a demo or contact us[62], and we'll be happy to assist.

Step 1: Installing & Starting IQ Server

Installing the IQ server is really a case of downloading the archived server, picking a location, and unpacking the contents. Since we won't be focused on mimicking a production experience, most laptop and desktop configurations should run IQ Server with no problem. If you are looking to plan for the future, be sure to review the Installation Requirements (see page 60).

1. Create an installation directory in your desired location.
2. Download the latest version of IQ Server to the installation directory.
3. Extract the `tar.gz` or `.zip` file.

Once you've extracted the contents, follow the steps below to run IQ Server:

1. Using a command line interface, switch to the `nexus-iq-server` bundle directory in your installation directory e.g. `nexus-iq-server-x.xx.x-xx-bundle`.
2. Run one of the following commands to start IQ Server:
   - Linux or Mac: `./demo.sh`
   - Windows: `demo.bat`
3. Open IQ Server in a browser using the default URL: http://localhost:8070[63]
4. Log in using the default Administrator account:
   - Username: `admin`
   - Password: `admin123`
5. Install the required product license supplied to you by the Sonatype Support team.
   - Click *Install License*.

62 mailto:support@sonatype.com
63 http://localhost:8070/

- Navigate to the license file (`.lic`) and click *Open*.
- Click *I Accept* to accept the End User License Agreement.

IQ Server needs access to an external data service to perform evaluations, which may be blocked in your internal environment. For a workaround, see Running IQ Server Behind a HTTP Proxy Server in IQ Server Configuration (see page 96).

Evaluating an application through the User Interface will transfer the bits to your IQ server. If you are working on a slower connection, or over a VPN, this mean longer analysis times.

Step 2: Importing Reference Policies

As of 1.44 (see page 22), the reference policy is automatically created for brand new IQ Server installations.

Policy is at the core of IQ Server's automation capabilities. This is true for both Nexus Firewall and Nexus Lifecycle. While you can create a completely custom set of policies, the Sonatype Reference Policy Set is the quickest way to get started. This set includes multiple policies for triggering violations on security vulnerabilities, licensing issues, architecture issues, and more.

The Reference Policy Set is downloaded and imported into the Root Organization automatically when IQ server is started for the first time.

You can also download the Reference Policy Set (see page 146) (`.json` file) and import it manually (see page 147).

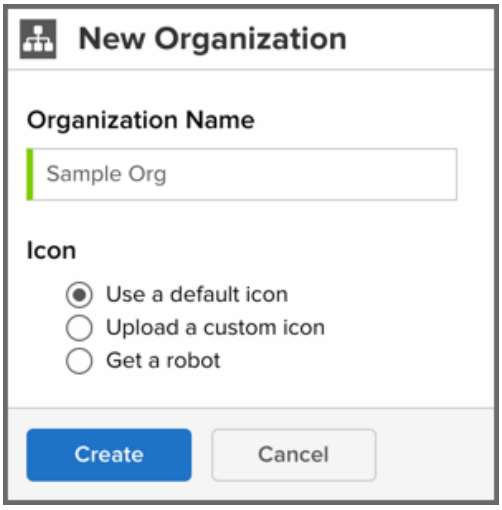
Step 3: Configuring Policy Actions

When evaluating applications, understanding IQ Server's system hierarchy is critical: Root Organization, organization, and application. This means policies and other configuration items are inherited from the Root Organization on down. This allows for easier policy management especially when you have multiple organizations and applications. Thus, in order to evaluate an application, you must have at least one organization and a corresponding application.

Creating an organization

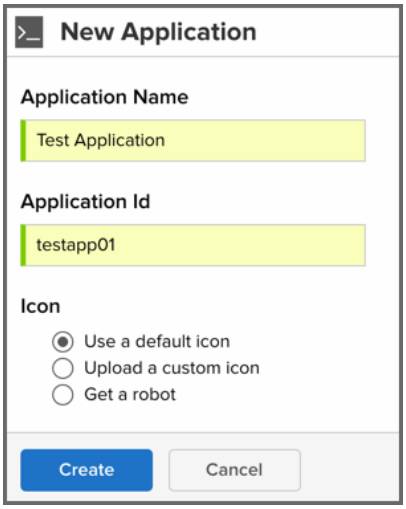
1. In the Organization & Policies area, with the Root Organization selected in the sidebar, click the *New Organization* button.
2. In the New Organization dialog, enter a name into the *Organization Name* text box.

3. Click the *Create* button.



Creating an application

1. With your newly created Organization selected in the sidebar, click the *New Application* button.
2. In the New Application dialog, enter an *Application Name* and *Application ID*.
3. Click the *Create* button.



Step 4: Evaluating Applications

After you install, start, and configure IQ Server, you are ready to evaluate applications. If you need a sample application, you can download WebGoat (`webgoat-container-x.x.x-war-exec.jar`) at https://github.com/WebGoat/WebGoat/releases.

To evaluate an application:

1. In the Organization & Policies area, select your application in the sidebar. The file that you evaluate will be associated with this application.

2. Go to the Actions menu, and click *Evaluate Binary*.
3. In the Evaluate a Binary dialog:
    a. Click the *Choose File* or *Browse* button, select the file to evaluate, and click *Open*.
    b. Select any stage to associate with the evaluation (e.g. Build).
    c. Click *No* to prevent sending notifications of policy violations as defined in the policy's configuration settings.
    d. Click the *Upload* button to begin evaluating the selected application.
4. When the evaluation is complete, click the *View Report* button to open the Application Composition Report for the application.

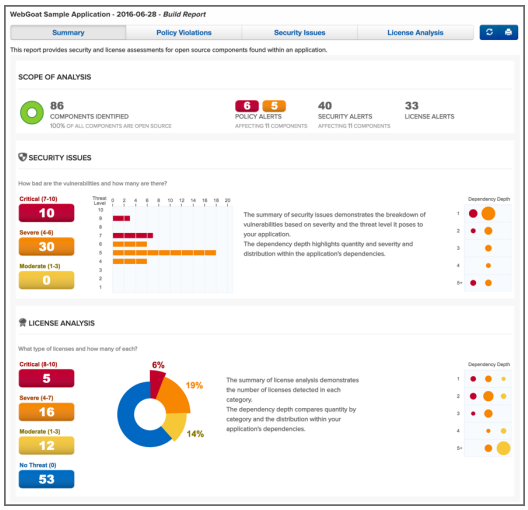
Step 5: Reviewing Results

Once evaluated, the results of a binary evaluation are displayed in the Application Composition Report, which you can always access by clicking the Reporting icon on the IQ Server toolbar.

The report's information is divided into four tabs:

- *Summary* - An overview of identified components and their policy alerts, security issues, and license analysis.
- *Policy Violations* - A list of violated policies and the components that triggered them sorted by threat level from from highest to lowest.
- *Security Issues* - A list of security vulnerabilities and the components that triggered them sorted by threat level from highest to lowest.
- *License Analysis* - A list of license issues and the components that triggered them sorted by license threat from highest to lowest.

For a more thorough explanation of the report, see Application Composition Report (see page 193).

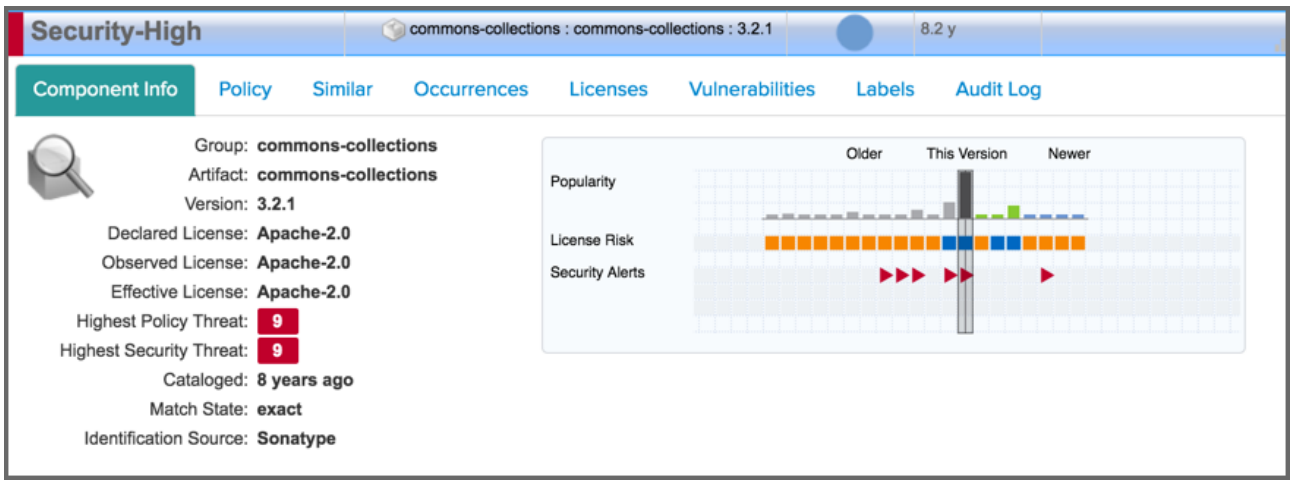


Step 6: Investigating & Remediating Violations

In the Application Composition Report, you can drill down to learn specific details about a violation. In every tab (except the Summary tab), you can click an individual component to open the Component Information Panel (CIP). The CIP displays many details, which are divided into different sections or tabs. To get you started using the CIP, take a look at these sections:

- *Component Info* - In the graph, you can move the vertical bar to learn the differences between versions of a component.
- *Policy* - You can click the Waive button to force IQ Server to ignore a policy violation.
- *Licenses* - You can track your research about a particular license and even override one.
- *Vulnerabilities* - You can click Info for a thorough explanation of a component's vulnerability and a recommended action.
- *Claim Component* - You can tell IQ Server to recognize a component even though it was previously identified as unknown.

This is just a small sample of the component information available in the CIP. For a complete discussion of the CIP, see Component Information Panel (see page 202).



Nexus Firewall Quick Start

This guide can help you get IQ Server up and running for the purpose of trying out the associated Nexus Firewall functionality before installing it in your development environment. If you have an available Nexus Repository Manager server available, you can expect to spend 15 to 30 minutes for installation and configuration, a bit longer if you don't.

To dive into Nexus Firewall a bit further, check out IQ Server and NXRM 3.x[64] or IQ Server and NXRM 2.x[65].

64 https://help.sonatype.com/display/NXI/IQ+Server+and+NXRM+3.x
65 https://help.sonatype.com/display/NXI/IQ+Server+and+NXRM+2.x

To integrate Nexus Repository Manager with IQ Server you need a Nexus Firewall license. If you don't have one, you can request a 14-day trial[66].

If you decide to use a trial license, we recommend installing a fresh copy of IQ Server instead of using an existing instance. This will simplify tear-down when your trial license expires.

Step 1: Installing & Starting IQ Server

Installing the IQ server is really a case of downloading the archived server, picking a location, and unpacking the contents. Since we won't be focused on mimicking a production experience, most laptop and desktop configurations should run IQ Server with no problem. If you are looking to plan for the future, be sure to review the Installation Requirements (see page 60).

1. Create an installation directory in your desired location.
2. Download the latest version of IQ Server to the installation directory.
3. Extract the `tar.gz` or `.zip` file.

Once you've extracted the contents, follow the steps below to run IQ Server:

1. Using a command line interface, switch to the `nexus-iq-server` bundle directory in your installation directory e.g. `nexus-iq-server-x.xx.x-xx-bundle`.
2. Run one of the following commands to start IQ Server:
   - Linux or Mac: `./demo.sh`
   - Windows: `demo.bat`
3. Open IQ Server in a browser using the default URL: http://localhost:8070[67]
4. Log in using the default Administrator account:
   - Username: `admin`
   - Password: `admin123`
5. Install the required product license supplied to you by the Sonatype Support team.
   - Click *Install License*.
   - Navigate to the license file (`.lic`) and click *Open*.
   - Click *I Accept* to accept the End User License Agreement.

IQ Server needs access to an external data service to perform evaluations, which may be blocked in your internal environment. For a workaround, see Running IQ Server Behind a HTTP Proxy Server in IQ Server Configuration (see page 96).

66 https://www.sonatype.com/firewall-for-oss
67 http://localhost:8070/

Evaluating an application through the User Interface will transfer the bits to your IQ server. If you are working on a slower connection, or over a VPN, this mean longer analysis times.

Step 2: Importing Reference Policies

As of 1.44 (see page 22), the reference policy is automatically created for brand new IQ Server installations.

Policy is at the core of IQ Server's automation capabilities. This is true for both Nexus Firewall and Nexus Lifecycle. While you can create a completely custom set of policies, the Sonatype Reference Policy Set is the quickest way to get started. This set includes multiple policies for triggering violations on security vulnerabilities, licensing issues, architecture issues, and more.

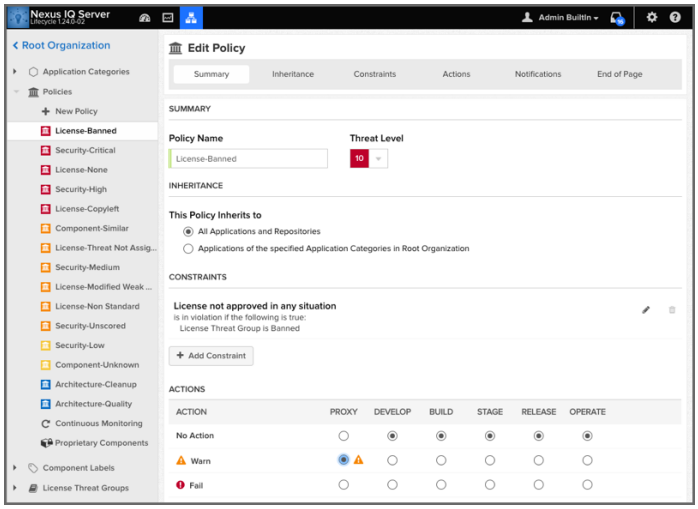
The Reference Policy Set is downloaded and imported into the Root Organization automatically when IQ server is started for the first time.

You can also download the Reference Policy Set (see page 146) (`.json` file) and import it manually (see page 147).

Step 3: Configuring Policy Actions

Policy Actions directly affect how IQ Server can automate processes in the available integrations when policy violations are encountered. In the case of Nexus Firewall, you can set an action to warn, which will audit, or simply display any violations. Alternatively you can set the action to Fail, which will quarantine, or block developers from accessing new components entering a repository that also violate the specified policy. To set Policy actions for the Proxy stage:

1. In IQ Server, click the *Organization & Policies* icon  on the IQ Server toolbar.
2. Click on the *Root Organization* in the sidebar, and then click the policies section.
3. Click on the policy you want to add an action to, and in the Proxy column choose *Warn* (Audit) or *Fail* (Quarantine).
4. Click the *Update* button.



⚠ When using the Fail action (Quarantine), the repository will need to be configured accordingly. A few additional things to keep in mind when using quarantine:

- New components entering the repository can be quarantined.
- Existing components in the repository will *not* be quarantined. This ensures that turning on quarantine will not break your existing builds.
- If IQ Server goes down or your license expires, you will *not* be able to proxy new components unless the IQ: Audit and Quarantine capability is disabled or deleted.
- If you delete the IQ: Audit and Quarantine capability, any quarantined components will be unquarantined.
- Developers will only see a 404 in their build logs when trying to retrieve a quarantined component, so it's important to let them know when quarantine is in use.

We recommend that you create a new proxy repository when trying out quarantine for the first time, for the best experience.

For additional information on what actions can be set and how they can affect automation, as well as the additional enforcement points supported by our Nexus Lifecycle product, be sure to check out Understanding the Parts of a Policy (see page 148).

Step 4: Nexus Repository Manager Configuration

IQ Server for Nexus Repository Manager allows you to integrate IQ Server's policy management and component intelligence features with proxy repositories in Nexus Repository Manager Pro. In order to do this, first you will need to configure the capabilities that allow for communication between IQ Server and Nexus Repository Manager. In addition, because Nexus Firewall is compatible with both Nexus Repository Manager OSS and Nexus Repository Manager Pro, there are specific instructions for each major version.

Supported versions

Nexus Repository Manager OSS: 2.14.8 or higher; 3.9.x or higher

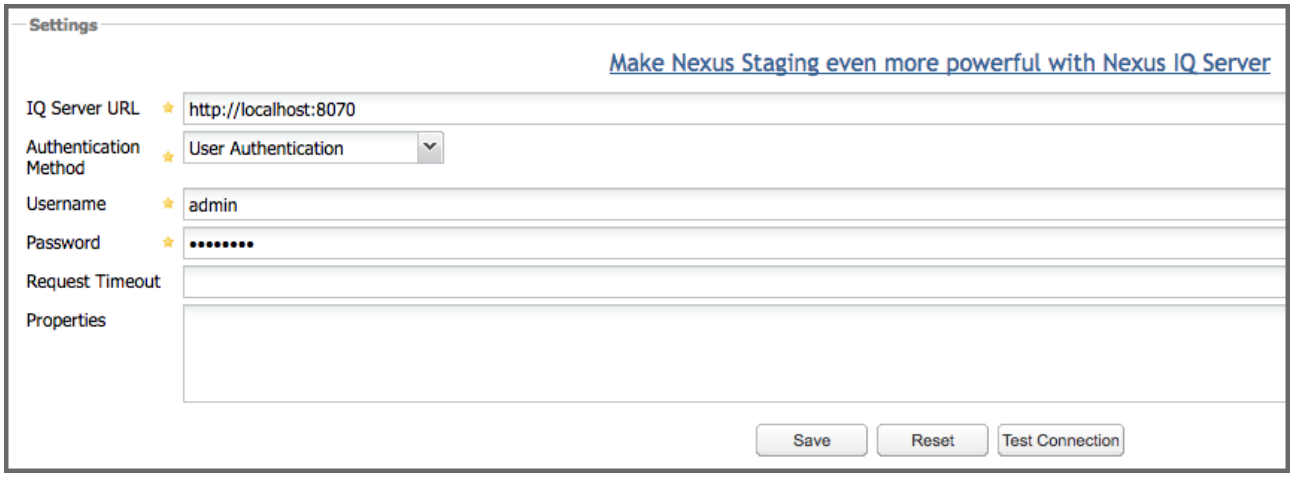
Nexus Repository Manager Pro: 2.12.x or higher; 3.2.x or higher

Nexus Repository Manager Pro with High Availability: 3.8.x or higher with Nexus IQ Server 1.35.x or higher

Configuring Nexus Repository Manager 2.x

There are two steps required for IQ Server to interact with an instance of Nexus Repository Manager and evaluate repositories. First, you need to configure the IQ Server connection:

1. In Nexus Repository Manager 2.x, click the *IQ Server Connection* menu item under Administration.
2. Enter the URL for your IQ Server installation.
3. Select an Authentication Method:
    a. User Authentication: Enter the username and password.
    b. PKI Authentication: Delegate to the JVM for authentication.
4. Click *Save*.



If successfully connected, a list of available applications in IQ Server displays in the Server Connection tab.

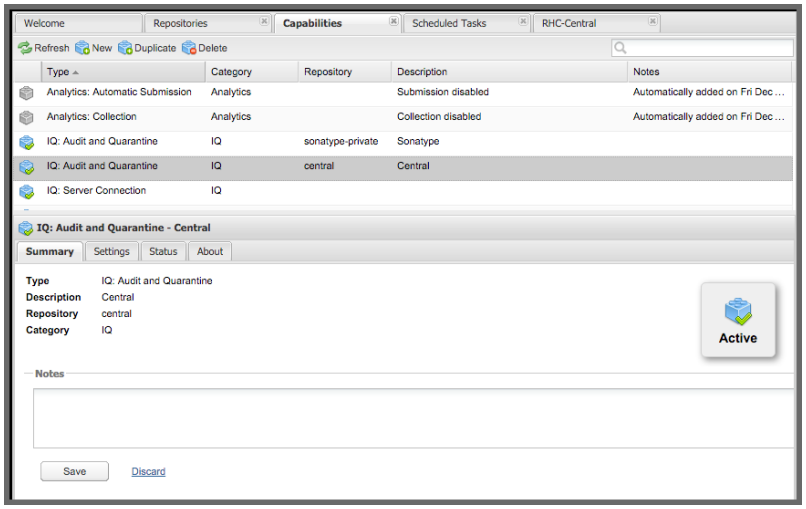
> For this quick start guide, using the default admin credentials is acceptable. However, for a real implementation, you would want to create a unique user for this integration, making sure to review Role Management (see page 121).

Next, add the Audit and/or Quarantine capability for each repository you want to evaluate. To configure Audit and/or Quarantine:

1. In Nexus Repository Manager, click *Capabilities* on the Administration menu.

2. Click the *New* button on the Capabilities tab. The Create new capability dialog is displayed.
3. In the Type list, choose *IQ: Audit and Quarantine*.
4. Select a specific proxy repository to analyze, for example Central.
5. Click *Add*.

An audit of the selected repository automatically starts. Nexus Repository Manager contacts IQ Server and evaluates the components within the selected repository against any associated policy.



These features use IQ Server policy management to identify, and if desired, prevent a proxy repository from serving unwanted components. If you have chosen to Audit, policies must also be configured with a fail action. For Quarantine configuration, see Configuring Audit and Quarantine[68]. Additional information is available in IQ Server and Repository Management[69].

Configuring Nexus Repository Manager 3.x

There are two steps required for IQ Server to interact with an instance of Nexus Repository Manager, and evaluate repositories. First, you need to configure the IQ Server connection:

1. In Nexus Repository Manager, click the *Administration* button on the main toolbar.
2. In the Administration main menu, click *Server* under IQ Server.
3. Select *Whether to use IQ Server* to enable IQ Server.
4. Enter the IQ Server URL.
5. Select an Authentication Method:
    a. User Authentication: Enter the username and password.

68 https://help.sonatype.com/display/NXI/IQ+Server+and+NXRM+2.x#IQServerandNXRM2.x-ConfiguringAuditandQuarantine
69 https://help.sonatype.com/display/NXI/IQ+Server+and+Repository+Management

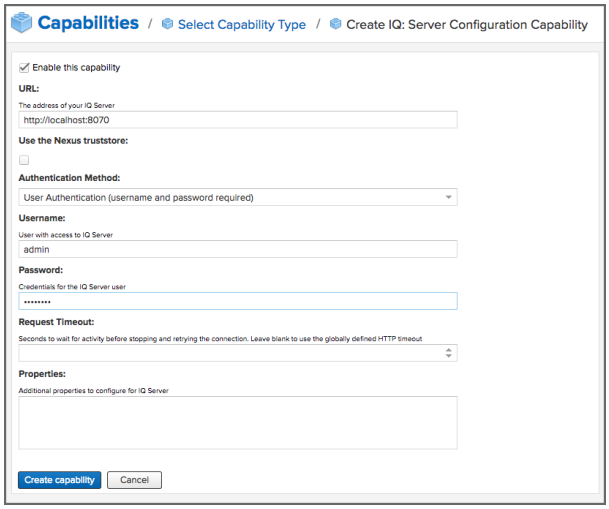
b. PKI Authentication: Delegate to the JVM for authentication.

6. Click *Verify Connection* to save.

> For this quick start guide, using the default admin credentials is acceptable. However, for a real implementation, you would want to create a unique user for this integration, making sure to review Role Management (see page 121).

Next, add the Audit and/or Quarantine capability for each repository you want to evaluate. To configure Audit and/or Quarantine[70]:

1. In Nexus Repository Manager 3.x, go to the *Administration* main menu and click *Capabilities* under System.
2. Click the *Create capability* button.
3. In the Select Capability Type view, click *IQ: Audit and Quarantine*.
4. Select a specific proxy repository to analyze, for example Central.
5. Click *Create capability* to save the new capability for Audit and Quarantine.



An audit of the selected repository is automatically started. Nexus Repository Manager contacts IQ Server and evaluates the components within the selected repository against any associated policy.

> These features use IQ Server policy management to identify, and if desired, prevent a proxy repository from serving unwanted components. If you have chosen to Audit, policies must also be

70 https://help.sonatype.com/display/NXI/IQ+Server+and+NXRM+3.x#IQServerandNXRM3.x-IntegratingNXRM3.xandIQServer-ConfiguringAuditandQuarantine

configured with a fail action. For Quarantine configuration, see Configuring Audit and Quarantine[71]. Additional information is available in IQ Server and Repository Management[72].

Step 5: Reviewing Repository Results

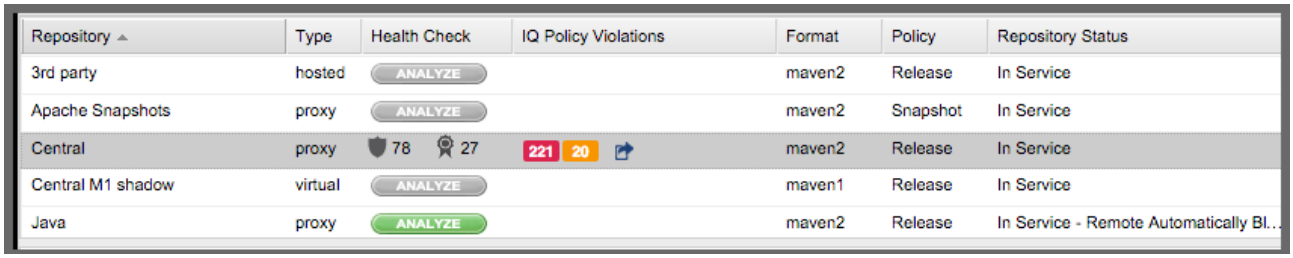
Once configured, the evaluation of the repository is automatic and will occur given any repository changes (e.g. adding a new component). Depending on the size (number of components) of the repository you configured, the evaluation could take a minute or so, but in general is very quick.

As you review the results, if you are not continuing on to review Nexus Lifecycle functionality, you can skip ahead to the investigation and remediation section, which provides additional details for drilling deeper into the results and available intelligence. Of course, a much more in-depth review of Nexus Firewall IQ Server can be found in IQ Server and Repository Results[73].

⚠ Accessing repository results will differ depending on the version of Nexus Repository Manager you have installed (differences highlighted below).

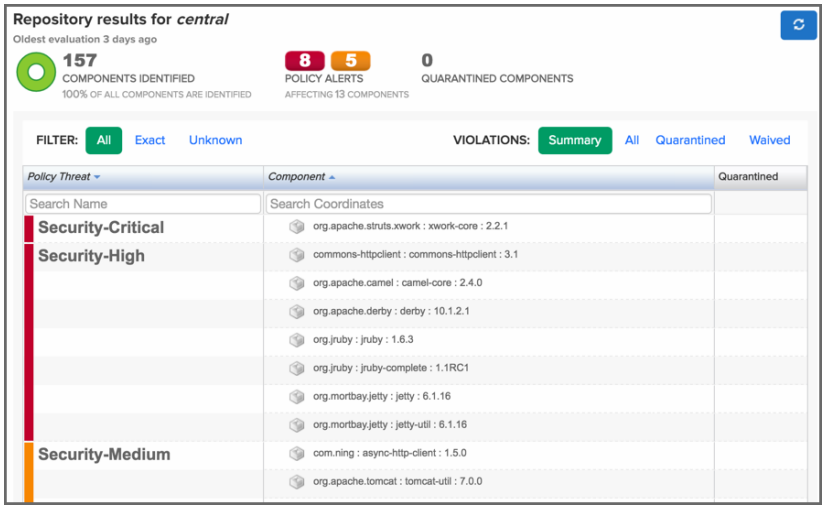
Reviewing Results in Nexus Repository Manager 2.x

To review results in Nexus Repository Manager 2.x, click *Repositories* under the Views/Repositories menu. Repository Results are summarized in the IQ Policy Violations column of the Repositories tab.



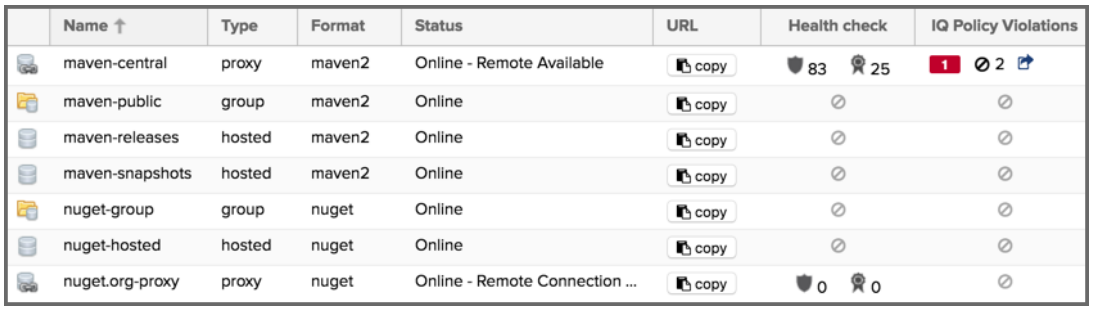
To view detailed results, click the open icon in the IQ Policy Violations column of the Repositories tab. IQ Server will open in a new tab showing detailed Repository Results.

71 https://help.sonatype.com/display/NXI/IQ+Server+and+NXRM+3.x#IQServerandNXRM3.x-ConfiguringAuditandQuarantine
72 https://help.sonatype.com/display/NXI/IQ+Server+and+Repository+Management
73 https://help.sonatype.com/display/NXI/IQ+Server+and+Repository+Results

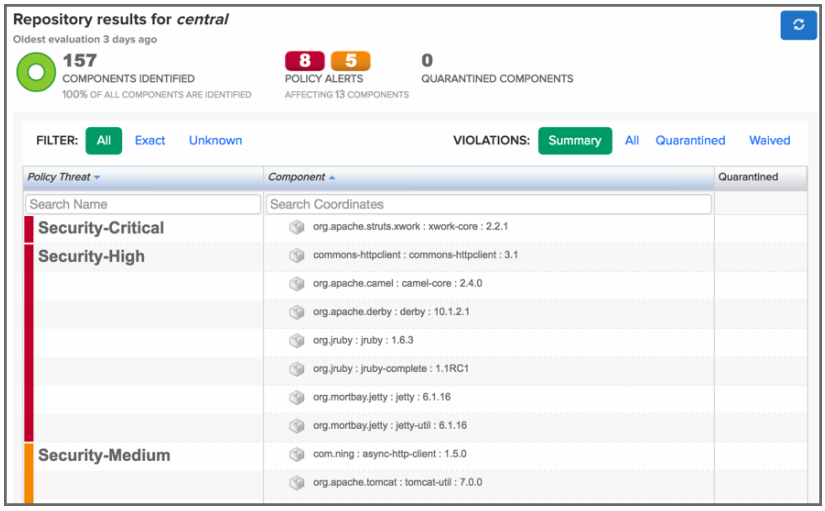


Reviewing Results in Nexus Repository Manager 3.x

In Nexus Repository Manager 3.x, the results of an audit are summarized in the IQ Policy Violations column of the Repositories view as shown in the figure below. Access the Repositories view from the Repository sub menu of the Administration menu.



To view detailed results, click the open icon in the IQ Policy Violations column of the Repositories view. IQ Server will open in a new tab showing detailed Repository Results.



Step 6: Investigating & Remediating Violations

Repository Results allow you to drill down to learn specific details about a violation, including the ability to isolate quarantined components. Click an individual component to open the *Component Information Panel* (CIP). The CIP displays many details, which are divided into different sections or tabs. To get you started using the CIP, take a look at these sections:

- *Component Info* - In the graph, you can move the vertical bar to learn the differences between versions of a component.
- *Policy* - You can click the Waive button to force IQ Server to ignore a policy violation.
- *Licenses* - You can track your research about a particular license and even override one.
- *Vulnerabilities* - You can click *Info* for a thorough explanation of a component's vulnerability and a recommended action.
- *Claim Component* - You can tell IQ Server to recognize a component even though it was previously identified as unknown.

This is just a small sample of the component information available in the CIP. For a complete discussion of the CIP, see Component Information Panel (see page 202).

Lifecycle XC

Overview

Lifecycle XC (Expanded Coverage) is a new capability of Nexus Lifecycle that utilizes OWASP dependency-check[74] to provide basic coverage for additional languages. Dependency-check is an open-source project used to scan applications and identify the use of known, vulnerable components. The focus of Lifecycle XC is not on being precise, but on leveraging a community project that lets us add basic coverage for a breadth of languages.

Nexus Lifecycle currently has advanced component intelligence for Java, JavaScript, and NuGet / .NET. The addition of Lifecycle XC brings basic-level intelligence to languages like Ruby, Swift, CocoaPods, and PHP.

> Lifecycle XC is supported in IQ Server versions 1.35 and newer.

74 https://jeremylong.github.io/DependencyCheck/index.html

What Does it Do?

Lifecycle XC provides basic coverage, including reporting vulnerabilities from the National Vulnerability Database (NVD), on languages we have not previously supported. Integrating dependency-check evidence collection allows us to quickly add languages as the community develops them, and provide a bill of materials for identified components and vulnerability data.

Lifecycle XC Does	Lifecycle XC Does Not
• Provide basic security data for vendor, product, and version (NVD only).	• Include Sonatype enhanced vulnerability information. • Perform policy evaluation. • Distinguish between external, proprietary, similar, and internally identified or claimed components. • Automatically analyze any dependencies. • Display available license information for any components. • Provide information via Webhooks or REST APIs.

How Does it Work?

The CLI for IQ Server provides the option to run in either normal (Lifecycle) mode or XC mode. The two modes are mutually exclusive. Data collection for identification and vulnerability correlation is provided by the dependency-check analyzers[75]. The results come from unverified public sources, and do not include any Sonatype enriched information. Policy evaluation is disabled due to the uncertainty of identification, quality of vulnerability information, and lack of additional information that is used in policy (e.g. license, popularity). Instead, XC results should be viewed as a bill of materials showing the component identified and the public vulnerability data.

⚠ Lifecycle XC (dependency-check) does not resolve dependencies. As a workaround, extract dependencies and place them in a location for scanning.

Use the following paramater in the CLI to run an xc scan:

```
-xc, --expanded-coverage
```

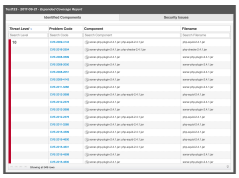
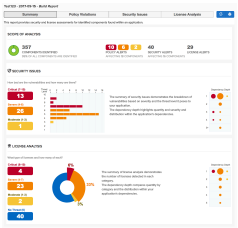
For an example XC evaluation, please see the Nexus IQ CLI[76] topic.

75 https://jeremylong.github.io/DependencyCheck/analyzers/index.html
76 https://help.sonatype.com/display/NXI/Nexus+IQ+CLI#NexusIQCLI-ExampleXCEvaluation

What Data Will I See?

When running the CLI in Lifecycle mode, you will continue to see enriched data and detailed reporting in the IQ Server for covered languages. When running the CLI in XC mode, you will see security data in an XC-branded report inside the IQ Server user interface. Lifecycle XC shows basic coverage information regardless if there is advanced support for the language/components being analyzed.

The table below shows an example of the data you will see when running the CLI in Lifecycle versus XC mode:

CLI MODE	LANGUAGE	PROCESSING	DERIVED DATA	EXAMPLE REPORT
XC	PHP	• Basic name-based matching • Vendor, product, and version evidence collection	• Name • CVE severity and CVE identifier	
Lifecycle	Java	• Precise fingerprint component matching • License analysis • Sonatype-enriched license research • Sonatype-enriched vulnerability research	• Identity • Age • Versions • Popularity • License data / metadata • Vulnerability data / metadata	

IQ Server Installation

Create the Installation Directory

⚠ Before installation, please check the IQ Server requirements (see page 60).

After a successful download (see page 53) of the IQ Server bundle archive, create an installation directory in the desired location and move the archive into the directory. For example:

```
cd /opt
mkdir nexus-iq-server
mv ~/Downloads/nexus-iq-server.* nexus-iq-server/
cd nexus-iq-server
```

Moving into the directory and extracting the archive with either one of these commands:

```
unzip nexus-iq-server*.zip
tar xfvz nexus-iq-server*.tar.gz
```

results in a directory with the following files:

```
README.txt
config.yml
demo.bat
demo.sh
eula.html
nexus-iq-server-1.27.0-01-bundle.tar.gz
nexus-iq-server-1.27.0-01.jar
```

Start the IQ Server

Once the IQ Server is installed, start it with:

```
cd /opt/nexus-iq-server
java -jar nexus-iq-server-*.jar server config.yml
```

This command starts the server with the IQ Server application using the configuration from the config.yml file and logging any output straight to the console. After a complete starts your console displays a message similar to:

```
... [main] org.eclipse.jetty.server.AbstractConnector - Started InstrumentedBlockingChannelConnector@0.0.
0.0:8070
... [main] org.eclipse.jetty.server.AbstractConnector - Started SocketConnector@0.0.0.0:8071
```

IQ Server now requires Java 8 starting with the 1.42 release. If you are running on an older version of Java (such as Java 7) you may encounter a message similar to the following on startup:

```
Exception in thread "main" java.lang.UnsupportedClassVersionError: com/sonatype/insight/brain/
service/InsightBrainService : Unsupported major.minor version 52.0
at java.lang.ClassLoader.defineClass1(Native Method)
at java.lang.ClassLoader.defineClass(ClassLoader.java:800)
at java.security.SecureClassLoader.defineClass(SecureClassLoader.java:142)
at java.net.URLClassLoader.defineClass(URLClassLoader.java:449)
at java.net.URLClassLoader.access$100(URLClassLoader.java:71)
at java.net.URLClassLoader$1.run(URLClassLoader.java:361)
at java.net.URLClassLoader$1.run(URLClassLoader.java:355)
at java.security.AccessController.doPrivileged(Native Method)
at java.net.URLClassLoader.findClass(URLClassLoader.java:354)
at java.lang.ClassLoader.loadClass(ClassLoader.java:425)
at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:308)
at java.lang.ClassLoader.loadClass(ClassLoader.java:358)
at sun.launcher.LauncherHelper.checkAndLoadMain(LauncherHelper.java:482)
```

If you encounter this message, you should upgrade to Java 8 to resolve the issue.

The command to start the server can be modified by adding java configurations parameters such as `-Xmx1024m` to improve performance and adapt to the server hardware.

At this stage you can access the web application at **port 8070** of your server via any web browser. Initial startup will display a screen for the License Installation.

We recommend always using SSL to ensure confidentiality of report and credential data during transit. This can be done by setting up a proxy server. You can find more details in the HTTPS Configuration section of IQ Server Configuration (see page 96).

Install the License

IQ Server requires a license to be installed. The required license file will be supplied to you by the Sonatype support team in the form of a `.lic` file.

Open a web browser and navigate to the IQ Server web application at port 8070 to install the license. Opening the URL, e.g. for a localhost deployment at http://localhost:8070, displays the Product License Configuration of the IQ Server.

Press the *Install License* button and select the `.lic` file in the file selector. Accept the end user license agreement to complete license installation. You will be taken to the *Getting Started* page that outlines initial steps towards getting value out of the IQ Server.

IQ Server Directories

When the IQ Server first starts, it creates a directory for the storage of all its data and configuration. This directory is configured in `config.yml` and defaults to `./sonatype-work/clm-server`. This path is relative to the location of the invoking java command.

Using the default startup command from the installation directory, causes `/sonatype-work/clm-server` to be created within it.

If you would like to separate the installation and data directories set `sonatypeWork` to a different location.

Additionally, a log directory is created within the installation directory. Use the `currentLogFilename` parameter in `config.yml` to change the location.

Related topics

- Automatic Shutdown on Errors (see page 85)
- Running IQ Server as a Service (see page 86)
- Backing up the IQ Server (see page 89)
- Upgrading the IQ Server (see page 90)

Automatic Shutdown on Errors

In rare circumstances, errors can occur that are severe enough as to be effectively unrecoverable under normal conditions. If this occurs, the system will shut down. This is intended to preserve a valid system state and avoid errant behavior or data corruption that could occur as a result of continuing in a potentially abnormal or undefined state.

For this reason, we recommend that you install the product in question as a service and ensure that the service will restart automatically in the event of a shutdown.

Nexus IQ Server

Nexus IQ Server has implemented this behavior since the 1.20 release. For this reason, we recommend that you configure Nexus IQ Server to run as a service (see page 86) and ensure that it restarts automatically to avoid a prolonged outage in the event of an unrecoverable error.

Running IQ Server as a Service

For production usage, it's strongly recommended to run the IQ Server as a service or daemon. This ensures that any operating system reboots include starting up the IQ Server.

A dedicated user for running a service is a well known best practice. This user should have reduced access rights as compared to the root user. Configuration of this user will depend on the operating system and security system used.

Once the user is configured, ensure that full access rights to the IQ Server installation directory are granted. An example command to achieve this for a service user with the username *iqserver* is:

```
chown -Rv iqserver /opt/nexus-iq-server
```

If the **`sonatypeWork`** parameter in **`config.yml`** points to a different directory, you must adjust the access rights for it as well.

The principal command for starting the IQ Server can be used in a simple startup script. Adjust the **`javaopts`** variable to suit the hardware used.

Startup Script

```
#! /bin/sh
cd /opt/nexus-iq-server
javaopts="-Xmx1024m -XX:MaxPermSize=128m"
java $javaopts -jar nexus-iq-server-*.jar server config.yml
A running server can be stopped with a simple shutdown script in Shutdown script.
```

Shutdown Script

```
#!/bin/sh
     pid=`ps aux | grep nexus-iq-server | grep -vE '(stop|grep)' | awk '{print $2}'`
     kill $pid
```

Typically these approaches are combined to a service script similar to the script listed in Simplistic Service Script for Unix Systems (see page 87). Saving this script as e.g. nexus-iq-server lets you to start the server with its log running to the current shell with:

```
./nexus-iq-server console
```

Starting as a background process can be initiated with:

```
./nexus-iq-server start
```

and running a background server can be stopped with:

```
./nexus-iq-server stop
```

This example script can be improved to be more robust against repeat invocations, long running stops, and potentially work better across different Unix flavors, but shows the principal functionality. A similar script can be used for Windows:

Simplistic Service Script for Unix Systems

```
#!/bin/sh

# The following comment lines are used by the init setup script like the
# chkconfig command for RedHat based distributions. Change as
# appropriate for your installation.

### BEGIN INIT INFO
# Provides:          nexus-iq-server
# Required-Start:    $local_fs $remote_fs $network $time $named
# Required-Stop:     $local_fs $remote_fs $network $time $named
# Default-Start:     3 5
# Default-Stop:      0 1 2 6
# Short-Description: nexus-iq-server service
# Description:       Start the nexus-iq-server service
### END INIT INFO

NEXUS_IQ_SERVER_HOME=/opt/tools/nexus-iq-server
VERSION=1.12.0
JAVA_OPTIONS="-Xmx1024m -XX:MaxPermSize=128m"
# The user ID which should be used to run the IQ Server
# # IMPORTANT - Make sure that the user has the required privileges to write into the IQ Server work
directory.
RUN_AS_USER=iqserver

do_start()
{
    cd $NEXUS_IQ_SERVER_HOME
    su -m $RUN_AS_USER -c "java $JAVA_OPTIONS -jar nexus-iq-server-$VERSION.jar server  config.yml > /dev/
null 2>&1 &"
    echo "Started nexus-iq-server"
}

do_console()
{
    cd $NEXUS_IQ_SERVER_HOME
    java $JAVA_OPTIONS -jar nexus-iq-server-$VERSION.jar server config.yml
}

do_stop()
{
```

```
    pid=`ps aux | grep nexus-iq-server | grep -vE '(stop|grep)' | awk '{print $2}'`
    kill $pid
    echo "Killed nexus-iq-server - PID $pid"
}

do_usage()
{
    echo "Usage: nexus-iq-server [console|start|stop]"
}

case $1 in
console) do_console
;;
start) do_start
;;
stop) do_stop
;;
*) do_usage
;;
esac
```

Setting the above as a startup script will vary between operating systems and distributions depending on the init system used. Generally the script would be copied to a dedicated startup directory and assigned with run-levels and other characteristics for the start up. The following commands show an example on a Debian-based system:

```
sudo su
cp nexus-iq-server /etc/init.d/
cd /etc/init.d
update-rc.d nexus-iq-server defaults
service nexus-iq-server start
```

Depending on the requirements from your system administrator, modify the scripts to fit into your environment and exact deployment scenario.

systemd

This example is a script that uses `systemd` to run the IQ Server. First create the nexus-iq-server script in `/etc/init.d` as described above. Then create a file called `nexusiq.service`. Add the following contents, and save the file in the `/etc/systemd/system/` directory:

```
[Unit]
Description=Nexus IQ Service
After=network.target

[Service]
Type=simple
ExecStart=/etc/init.d/nexus-iq-server start
```

```
ExecStop=/etc/init.d/nexus-iq-server stop
User=iqserver
Restart=on-abort

[Install]
WantedBy=multi-user.target
```

Activate the service with the following commands:

```
sudo systemctl daemon-reload
sudo systemctl enable nexusiq.service
sudo systemctl start nexusiq.service
```

Note that systemd sets the user ID as configured in the "User" property. This means you need to modify the service script so that it does not run an "su" command, since that will not work when run as the non-root user.

To do this change this line:

```
su -m $RUN_AS_USER -c "java $JAVA_OPTIONS -jar nexus-iq-server-$VERSION.jar server  config.yml > /dev/null 2>&1 &"
```

To this:

```
java $JAVA_OPTIONS -jar nexus-iq-server-$VERSION.jar server  config.yml
```

Our support team can assist you with operating system and Linux distribution-specific tips and tricks regarding the startup script and installation.

Backing up the IQ Server

We highly recommend that you establish a data recovery plan in accordance with your company's policies.

The IQ Server keeps all its configuration and data, besides the startup configuration, in the **`sonatypeWork`** folder as configured in **`config.yml`**. By default, this folder will be the **`/sonatype-work/clm-server`** folder in your installation directory.

There is a tight coupling between report data stored in the file system and the data stored in the H2 database. If the IQ Server is running, any backup strategy for **`sonatypeWork`** runs the risk of an open database and may lead to a corrupt backup. To avoid this, the IQ Server should be shut down before performing the backup.

Upgrading the IQ Server

The latest version of IQ Server can be downloaded from the IQ Download and Compatibility (see page 53) page.

Before starting any upgrade, make sure you have reviewed all upgrade instructions provided for your current version (see additional sections below), as well as any versions that followed.

To upgrade the IQ Server:

1. Stop the IQ Server
2. Perform a backup (see page 89)
3. Copy the new jar and change the startup scripts to reflect the new jar name
4. Start the IQ Server

⚠ Optionally you can manually merge the changes from the existing **`config.yml`** into the new, but this is not required.

If there is any concern, please feel free to contact our support team: support@sonatype.com[77].

Upgrading from Version 1.44 or Earlier to Version 1.45 or Later

Version 1.45 of IQ Server introduces a more compact format to store the policy violation data of your applications. Upgrading to this version and the new storage format can take notable time. To properly prepare for this upgrade, refer to our detailed instructions on Upgrading the IQ Server to Version 1.45 (see page 93).

Upgrading from Version 1.42 or Earlier to Version 1.43 or Later

IQ Server version 1.43 uses a more powerful configuration format (config.yml).

⚠ If you wish to use a configuration file from a prior version, then you must update it. Please refer to our configuration update guide (see page 107) for more information.

77 mailto:support@sonatype.com

Upgrading to Version 1.42 or Later requires Version 1.16 or Later

IQ Server version 1.42 or later will no longer perform data migrations for versions prior to or including 1.16.

This is in an effort to streamline future data migrations to improve data storage and its scalability.

Upgrading from Version 1.35 or Earlier to Version 1.36 or Later

IQ Server version 1.36 replaces the *Security Vulnerability present* policy condition with the *Security Vulnerability Severity greater than or equal to 0* policy condition and removes the *Security Vulnerability absent* policy condition. The upgrade to 1.36 or later will fail if you have any policies relying on the *Security Vulnerability absent* policy condition. Before attempting to upgrade, we highly recommend that you perform a backup (see page 89). If the upgrade fails, then you can still start the previous version against this backup (see page 89). In the meantime, you can contact our support team: support@sonatype.com[78] or your customer success representative directly for assistance in changing any of your policies that rely on the *Security Vulnerability absent* policy condition.

Upgrading from Version 1.17 or Earlier to Version 1.18 or Later

IQ Server version 1.18 introduces the Root Organization—a new entity at the top of the system hierarchy that allows you to set policy globally across all organizations and applications. After you update the IQ Server to version 1.18, you should configure and create the Root Organization. It's a one time process, and occurs when the server is restarted. The process makes a permanent change to the system hierarchy that cannot be undone. It is strongly recommended that you backup the IQ Server and read "Introducing the Root Organization" before proceeding.

In IQ Server version 1.21, the Sonatype CLM for Hudson and Jenkins plugin has been updated and rebranded to Nexus IQ for Hudson/Jenkins 1.x. If you have a prior version of the plugin installed, then you must uninstall the older version before installing the newer rebranded one. For installation instructions, see the Nexus IQ for Hudson/Jenkins 1.x chapter.

IQ Server version 1.26 introduces CSRF protection for all available plugins that use reverse proxy authentication. This new protection is enabled by default. If you want to upgrade to IQ Server version 1.26 and use reverse proxy authentication in your plugins, you should upgrade your plugins to their latest versions first.

⚠ If you would like to upgrade to IQ Server version 1.26 and use reverse proxy authentication with older plugin versions, you will need to disable CSRF protection for reverse proxy authentication. See the section on Reverse Proxy Authentication for more information.

78 mailto:support@sonatype.com

⚠ Both Nexus Repository Manager version 2.14.3 and older and Nexus Repository Manager version 3.2.1 and earlier 3.x versions do not support CSRF protection when using reverse proxy authentication. If you want to use reverse proxy authentication with these Nexus Repository Manager versions and IQ Server 1.26 or later, you will need to disable CSRF protection for reverse proxy authentication. See the section on Reverse Proxy Authentication for more information.

Upgrading from Version 1.15 or Earlier to Version 1.23 or Later

Due to data migrations, you will need to upgrade to version 1.16 first before proceeding to upgrade to version 1.23 or later versions of IQ Server.

Upgrading from Version 1.16 or Earlier

In version 1.17 a rebranding of the Sonatype CLM product took place, and is now known as Nexus IQ. As part of this rebranding two of the binaries also changed during this release:

Server

- From: sonatype-clm-server
- To: nexus-iq-server

CLI

- From: sonatype-clm-scanner
- To: nexus-iq-cli

If you have any scripts utilizing the previous names, you will want to update these given the change above.

⚠ In the example above, only the server name is given. The full binary name would look like nexus-iq-server-1.27.0-01.jar

Upgrading from Versions Earlier than 1.9.x

While Sonatype only supports the previous two releases, we are happy to help direct any upgrade needs you may have. If you are upgrading from a version prior to 1.9.x, please contact our support team directly: support@sonatype.com[79].

79 mailto:support@sonatype.com

Upgrading the IQ Server to Version 1.45

If you have been directed here from your IQ Server's error log and this is the first time you see these upgrade instructions, we suggest you abort the server upgrade you were about to commence and revert to your previous IQ Server version.

The upgrade procedure outlined below requires preparation and should not be rushed. Read these instructions carefully and only retry the server upgrade once you have had a chance to make the necessary preparations.

Starting with version 1.45 or greater, IQ Server uses a more effective and condensed format to store policy violation data. Depending on the database size of your installation and the hardware you use, upgrading your database to this new format can take notable time (up to hours in worst case scenarios). These instructions are meant to help you plan and reduce the downtime of your IQ Server while the upgrade is carried out.

Preparing for the Upgrade

Upgrade to IQ Server 1.44 First

If your current installation is several versions behind and not already running IQ Server version 1.44 you should update to that version first. This reduces the number and duration of upgrade tasks both the server and its administrator have to consider when later upgrading to version 1.45 or greater. In particular, you should perform the update to version 1.44 during a different maintenance window than the update to version 1.45 or greater.

Version 1.44 can be obtained from these links

- nexus-iq-server-1.44.0-01-bundle.tar.gz[80]
- nexus-iq-server-1.44.0-01-bundle.zip[81]

Ensure Sufficient Free Disk Space

Until the upgrade is fully completed, the IQ Server's database will temporarily contain policy violation data in both the old and new storage format. As a result, the database will generally grow while the upgrade is in progress. To avoid running out of disk space during the upgrade, we recommend having at least as much free space as the database currently occupies on the disk hosting it. You can determine the amount needed by looking at how large the database file is, e.g. `sonatype-work/clm-server/data/ods.h2.db` (see your server's `config.yml` and its `sonatypeWork` option for the path applicable to your installation).

---

80 https://download.sonatype.com/clm/server/nexus-iq-server-1.44.0-01-bundle.tar.gz

81 https://download.sonatype.com/clm/server/nexus-iq-server-1.44.0-01-bundle.zip

Perform a Test Run of the Upgrade to 1.45 or greater in a Staging Environment

Given the number of unknown variables about your particular installation, it is impossible to predict the duration of the upgrade and the maintenance time you should expect. By performing a test run of the upgrade in a non-production environment, you can learn about its approximate duration and plan your maintenance for the real upgrade accordingly.

For that test run to be representative, it should be executed using a recent backup/clone of your installation and on hardware closely comparable to your production system. Especially the I/O performance of the storage holding the server's database (`sonatype-work/clm-server/data`) is a key factor for the upgrade duration.

The test run also helps you estimate the resulting database size and verify that your storage system has enough free space to accommodate it.

Compact the Database Before the Upgrade

A long-lived database contains some percentage of empty space to support its efficient growth. If your server's database (`sonatype-work/clm-server/data/ods.h2.db`) is several gigabytes large, this percentage of empty space can amount to gigabytes as well. Removing this empty space, a process called compacting, can reduce the upgrade duration for IQ Server 1.45 or greater.

To compact your server's database:

1. Shut down your IQ Server
2. Execute the command `java -jar nexus-iq-server-1.44.0-01.jar` **`compact-db`** `config.yml` or equivalent for your environment
3. You may restart your IQ Server once the above command has finished

Compacting the database is an I/O heavy operation and, depending on the size of your database and the storage performance, can take several minutes or longer. To help plan the needed maintenance window, we suggest you perform a test run of this process in a staging environment on comparable hardware using a recent backup of your installation.

Over time and with continued use of IQ Server, the database will again accumulate some empty space. In order to benefit the upgrade to IQ Server 1.45 or greater, compacting the database should therefore be done not too far in advance of the upgrade, e.g. within a week prior to the 1.45 or greater upgrade. If easier for you to implement, you can perform this step right before upgrading to version 1.45 or greater. We primarily mention it as a separate preparation step to allow you to organize the upgrade over several smaller maintenance windows instead of one long maintenance if desired.

Use a Fast Disk/Storage

The upgrade to IQ Server 1.45 or greater is an I/O intensive task. The performance of the storage device holding the server's database (`sonatype-work/clm-server/data`) is a key factor for the upgrade duration.

Depending on your environment, it can be beneficial to at least temporarily employ a faster disk for IQ Server and its upgrade.

Internal Reference Numbers

Our internal benchmarks may provide some rough guidance on expected upgrade times, though as noted above the upgrade complexity is highly dependant on your particular installation.

Database Size	Upgrade Time SSD	Upgrade Time HDD
3.3 GB	~ 1 minute	~ 10 minutes
7.2 GB	~ 2 minutes	~ 15 minutes
8.9 GB	~ 3 minutes	~ 40 minutes
19.0 GB	~ 5 minutes	~ 60 minutes
67.0 GB	~ 60 minutes	~ 15 hours

Performing the Upgrade

Take a Fresh Backup Before the Upgrade

This should be second nature for the diligent reader of our general upgrade instructions (see page 90) but is worth emphasizing: Backing up the IQ Server (see page 89) before the upgrade ensures you have something to fall back to in the unlikely event something goes haywire.

Update the Server's `config.yml` to Consent to the Upgrade

To save unsuspecting administrators from surprises, IQ Server 1.45 or greater requires a change to the configuration file to signal the implications of the upgrade are understood and the necessary preparations have been made. So for the aforementioned test runs and the actual upgrade of your production system, the following line needs to be inserted into the `config.yml` file used to launch IQ Server 1.45 or greater:

```
# Confirm you have read the upgrade instructions and come prepared
consentToUpgradeToVersion_1_45: true
```

If the line is missing, IQ Server will refuse to start with an error in its log, directing to these upgrade instructions.

Once the upgrade to version 1.45 or greater has completed, you can remove that line from the `config.yml` again.

Compact the Database After the Upgrade

Once IQ Server has completed the upgrade to 1.45 and is up and running again, its database will still roughly have the same size as before. To cash in on its more efficient storage format for policy violation data, its database needs to be compacted (again) as described earlier. This task can be done at a later time like the next available maintenance window or right after the upgrade, whatever works best with your maintenance schedule.

To compact your server's database:

1. Shut down your IQ Server
2. Execute the command `java -jar nexus-iq-server-1.45.0-01.jar compact-db config.yml` or equivalent for your environment or version.
3. You may restart your IQ Server once the above command has finished

You might choose to skip compacting the database before the upgrade, e.g. because your test run found the upgrade was already sufficiently quick, but we strongly recommend to compact the database after the upgrade to 1.45 or greater. In our testing, we observed size reductions of 50% and more.

IQ Server Configuration

Overview

The IQ Server is an application exposed using a Dropwizard[82] server.

The main configuration file for the IQ Server installation is a YAML formatted[83] file called **`config.yml`** found in the installation directory. The config.yml file typically contains only those configuration options which are rarely changed.

Special considerations when editing the config.yml file:

- TAB characters are not supported, use space characters only for indenting
- structure is tree-like - indents define structure hierarchy and are relevant to proper parsing of the file
- indented lines are considered child options of the first un-commented outdented line preceeding them
- commented lines are ignored - comments begin with the **#** character
- an improperly formatted config.yml will prevent the server from starting

We strongly recommend using a text editor that will inform you of any TAB characters accidentally inserted into the file.

82 http://www.dropwizard.io/
83 https://en.wikipedia.org/wiki/YAML#Syntax

Options that are more commonly changed are typically found in the *System Preferences* section of the IQ Server user interface, which you can access by clicking on the *System Preferences* icon located in the top right of the IQ Server header ( ).

Configuring Outbound Traffic

Network Access to Sonatype Data Services

The IQ Server needs to communicate securely with the Sonatype Data Services using HTTPS.

Firewall and HTTP proxy server administrators must ensure the following URL is accessible from the Nexus IQ Server process:

`https://clm.sonatype.com:443`

The Nexus IQ Server may also be configured to send HTML based notification emails to your users. These emails contain links to static resources loaded from:

`http://cdn.sonatype.com:80`

Therefore, email clients which load notification messages should have access to the **cdn.sonatype.com** sub-domain to ensure complete HTML formatted rendering.

HTTP Proxy Server

Many organizations filter, control and optimize HTTP network traffic via an HTTP proxy server[84].

To allow the IQ Server to reach Sonatype Data Services, you may have to configure IQ Server to use a specific HTTP Proxy Server for outbound requests. The proxy server must support the CONNECT method[85] of tunneling. The connection details are specified in the `proxy` section of the `config.yml` file, which by default is commented out.

Example Disabled Proxy Configuration in config.yml

```
# Proxy settings.
#proxy:

  # The host running the proxy server to use.
  #hostname: "127.0.0.1"

  # The port at which the proxy server listens on.
  #port: 80
```

84 https://en.wikipedia.org/wiki/Proxy_server
85 https://en.wikipedia.org/wiki/HTTP_tunnel#HTTP_CONNECT_method

```
  # The username used to access the proxy server.
  #username: "anonymous"

  # The password used to access the proxy server.
  #password: "guest"
```

Uncomment the proxy section and adjust the values to match your configuration.

Example Enabled Proxy Configuration in config.yml

```
# Proxy settings.
proxy:

  # The host running the proxy server to use.
  hostname: "http-proxy-host.example.com"

  # The port at which the proxy server listens on.
  port: 8888

  # The username used to access the proxy server.
  #username: "anonymous"

  # The password used to access the proxy server.
  #password: "guest"
```

NTLM Authentication

If your proxy server uses NTLM authentication supply your user name in the following format:

Example NTLM Authentication Based HTTP Proxy Username

```
  username: "DOMAIN\\username"
```

Appending a User Agent To Outbound Requests

To address the firewall configurations set by some organizations, you can customize the user agent header used for HTTP requests. To add a user agent string, add the following line to the IQ Server config.yml:

```
userAgentSuffix: "test string"
```

Control characters are not permitted in the user agent and the max length of the text is 128 characters.

Configuring Inbound Traffic

HTTP Configuration

The port parameter(s) in the IQ Server `config.yml` allow you to set the port(s) to access the application and/or operational menu. Each port can be freely changed to other values, as long as it is not used and in the allowed range of values greater than 1024. The following examples show how to set these port parameter(s).

For IQ Server version 1.42 and lower

HTTP Configuration in config.yml

```
http:
    port: 8070
    adminPort: 8071
    adminUsername: user1234
    adminPassword: pass5678
```

Access to the operational services available on the `adminPort` can optionally be restricted with HTTP basic authentication by specifying `adminUsername` and `adminPassword`.

Note that `adminUsername` and `adminPassword` are only available in IQ Server version 1.42 and lower. Additionally the credentials are vulnerable to a timing attack, see CVE-2017-9735[86] for details. In environments where this risk is not tolerable, please use a reverse proxy instead to shield and authenticate access to the operational services.

For IQ Server version 1.43 and higher

HTTP Configuration in config.yml

```
server:
    applicationConnectors:
        - type: http
          port: 8070
    adminConnectors:
        - type: http
```

86 https://nvd.nist.gov/vuln/detail/CVE-2017-9735

```
        port: 8071
```

HTTPS/SSL Configuration

One option to expose the IQ Server via https, is to use an external server like Apache httpd or nginx and configure it for reverse proxying the external connections via https to internal http connection. This reverse proxy can be installed on the same server as the IQ Server or a different server and numerous tutorials for this setup are available on the internet.

A second option is to directly configure SSL support for Dropwizard by modifying the relevant segment in the `config.yml` file. The following examples show how to do this. Note that the keystore file can be generated and managed with the keytool[87].

For IQ Server version 1.42 and lower

HTTPS Configuration in config.yml

```
http:
  port: 8443
  adminPort: 8471

  connectorType: nonblocking+ssl

  ssl:
    keyStore: /path/to/your/keystore/file
    keyStorePassword: yourpassword
```

Further documentation is available in the old Dropwizard documentation[88].

For IQ Server version 1.43 and higher

HTTPS Configuration in config.yml

```
server:
    applicationConnectors:
        - type: https
          port: 8443
          keyStorePath: /path/to/your/keystore/file
          keyStorePassword: yourpassword
    adminConnectors:
        - type: https
          port: 8471
```

87 https://docs.oracle.com/javase/7/docs/technotes/tools/windows/keytool.html
88 http://dropwizard.github.io/dropwizard/0.6.2/manual/core.html#ssl

```
        keyStorePath: /path/to/your/keystore/file
        keyStorePassword: yourpassword
```

Further documentation is available in the new Dropwizard documentation[89].

Web Application Context Path

For IQ Server 1.43 and newer the context path at which the web application is accessible can be customized using the option shown below:

```
server:
    # The context path for the application. Note that this must have a leading slash.
    applicationContextPath: /
```

CSRF Protection

Attacks on the IQ Server could occur via a cross-site request forgery (CSRF). To protect against this, a configuration item `csrfProtection` has been provided. This option is set to `true` by default.

```
# Enables/disables cross-site request forgery protection. Defaults to true for increased security.
#csrfProtection: true
```

⚠ In cases where the HTTP headers are stripped (e.g. a proxy configuration), this protection would block usage of the UI. To address this, you can disable this protection by setting the configuration item to false.

Security

Anonymous Access

By default the IQ Server requires users to authenticate when submitting applications for evaluation. While not recommended, if you need to allow anonymous application evaluation submissions, add the following line to the config.yml:

```
anonymousClientAccessAllowed: true
```

89 http://www.dropwizard.io/1.2.2/docs/manual/core.html#ssl

Reverse Proxy Authentication

Browser-based single sign-on (SSO) configurations allow a user to log into the system in a web browser without the need to log into any individual web applications. Any user navigation to further applications carries the authenticated username through to the application and the user is automatically logged in.

Typically this is implemented with a reverse proxy server and the username is supplied via a HTTP header field.

The IQ Server can be configured to accept this kind of SSO configuration in the config.yml file, allowing you to specify the exact header field to be used:

```
# Configures reverse proxy authentication for the web UI.
reverseProxyAuthentication:
    # Set to true to activate authentication
    enabled: true
    # Name of the HTTP request header field that carries the username
    usernameHeader: "REMOTE_USER"
    # Set to true for backward compatibility with old client plugins
    csrfProtectionDisabled: false
        # The service URL that will be redirected to when a user requests logout.
        logoutUrl: http://localhost/logout/index.html
```

⚠ When using reverse proxy authentication from integration points to IQ Server, Cross-Site Request Forgery (CSRF) protection is enabled by default. If an integration does not support CSRF protection, it should be updated to the latest version. Alternatively, CSRF protection can be disabled by setting *csrfProtectionDisabled: true* in the IQ Server configuration.

⚠ The `logoutUrl` property of the `reverseProxyAuthentication` configuration is only supported from the IQ Server version 1.35.0 and higher.

The default `config.yml` contains a commented out section for this configuration with some further details.

This authentication method applies to all users, both IQ Server and LDAP users. Incoming usernames are matched first to IQ Server users, then to LDAP users, and then the configuration in the IQ Server determines the access level granted to the user.

Public Key Infrastructure (PKI) Authentication

⚠ In order to implement PKI authentication, a reverse proxy server is needed to translate PKI supplied credentials to users known by IQ Server.

Tools and plugins can be configured to use PKI authentication, which delegates authentication to the Java Virtual Machine (JVM). When delegated, the tool or plugin does not handle authentication and instead the JVM supplies PKI information to the reverse proxy for authentication.

For information on setting PKI authentication for a specific tool or plugin, please see the appropriate page.

File Configuration

IQ Server stores various files and data related to its operations in a work directory. By default this data is stored in a /sonatype-work/clm-server directory in the path the server runs. The directory is configurable using the `sonatypeWork` field in File Configuration in config.yml.

File Configuration in config.yml

```
sonatypeWork: ./sonatype-work/clm-server
```

In addition, IQ Server uses the system temporary directory during its operation. This folder varies by operating system but is usually controlled by an environmental variable. If a specific directory needs to be used, the IQ Server can be started with a command line flag as such:

```
cd /opt/nexus-iq-server
java -jar -Djava.io.tmpdir=/path/to/tmpdir nexus-iq-server-*.jar server config.yml
```

Note that the user account which the server runs under must have sufficient access rights to both the work and temporary directory in order for IQ Server to function properly.

Email Configuration

Email Server

The IQ Server can be configured to send email notifications for events such as policy violation notifications. This functionality requires an SMTP server, which is configured along with a number of other options in the mail section of the config.yml file displayed in Mail Configuration in config.yml.

Mail Configuration in config.yml example

```
mail:
    hostname: your.mailserver.com
    port: 465
    username: user@company.com
    password: password
    tls: true
    ssl: true
    systemEmail: "Sonatype@localhost"
```

The connection details are established with `hostname` and `port` and optionally with the addition of `username`, `password`, `tls` and `ssl`. The `systemEmail` parameter will be used as the sender email for any emails the IQ Server sends. All fields are required.

Finally, when setting email configuration, make sure you have also set the `Base URL`, otherwise sending of notification emails may fail.

Setting the Base URL

The *Base URL* is the web address used when navigating to the server user interface or using the REST API. **The IQ Server only uses this value to construct absolute URLs to your user interface inside of email notifications**. The most common reason why the address would be different is if you have a reverse proxy that terminates HTTP requests at an address different from where the IQ Server is running.

To configure the `baseUrl` parameter, uncomment the baseUrl setting in config.yml

Example Enabled baseUrl

```
baseUrl: http://nexus-iq-server.example.com/
```

IQ Server uses the `X-Forwarded-Proto` and `X-Forwarded-Host` headers to resolve user-facing URLs when a HTTP request comes through a reverse proxy server. Please refer to the documentation of your proxy server to correctly configure it to set these headers.

Logging Configuration

A `log` directory is created under the installation directory and used to store all log files. Logs are rotated into 50 days worth of gzipped dated archives.

Application Log

The last active application log is found at `./log/clm-server.log` and archived logs are compressed with name pattern of `clm-server-yyyy-MM-dd.log.gz`. The application log can be customized in the `logging` section of the `config.yml` file.

IQ Server 1.43 and newer uses Dropwizard 1.2.2 logging configuration[90].

IQ Server 1.42 and older uses Dropwizard 0.6.2 configuration[91].

Request Log

The last active request log is found at `./log/request.log` and archived logs are compressed with the name pattern of `request-yyyy-MM-dd.log.gz`. The request log can be customized in the `requestLog` section of the `config.yml` file.

IQ Server 1.43 and newer uses Dropwizard 1.2.2 requestLog configuration[92].

IQ Server 1.42 and older uses Dropwizard 0.6.2 configuration[93] .

Request Log Line Formats

As of IQ Server 1.43 the request log line format is now configurable using patterns defined by the Logback Access PatternLayout[94] class.

The **Sonatype preferred format** renders timestamps with the same timezone as the application log to ease comprehension during comparison.

Preferred request.log line format

```
logFormat: "%clientHost %l %user [%date] \"%requestURL\" %statusCode %bytesSent %elapsedTime
\"%header{User-Agent}\""
```

Concise config.yml example of specifying request.log line format

```
server:
  requestLog:
```

90 http://www.dropwizard.io/1.2.2/docs/manual/core.html#id4
91 http://www.dropwizard.io/0.6.2/manual/core.html#id5
92 http://www.dropwizard.io/1.2.2/docs/manual/configuration.html#request-log
93 http://www.dropwizard.io/0.6.2/manual/core.html#id5
94 https://logback.qos.ch/xref/ch/qos/logback/access/PatternLayout.html

```
    appenders:
      - type: file
        logFormat: "%clientHost %l %user [%date] \"%requestURL\" %statusCode %bytesSent %elapsedTime
\"%header{User-Agent}\""
        currentLogFilename: ./log/request.log
        archivedLogFilenamePattern: ./log/request-%d.log.gz
        archivedFileCount: 50
```

IQ Server 1.45 and newer default request.log line format if unspecified

```
logFormat: "%clientHost %l %user [%date] \"%requestURL\" %statusCode %bytesSent %elapsedTime
\"%header{User-Agent}\""
```

IQ Server 1.43 and 1.44 default request.log line format if unspecified

```
logFormat: "%clientHost %l %user [%t{dd/MMM/yyyy:HH:mm:ss Z,UTC}] \"%requestURL\" %statusCode %bytesSent
\"%header{Referer}\" \"%header{User-Agent}\" %elapsedTime"
```

IQ Server 1.42 and earlier request.log line format with simulated dispatch time

```
# specifying log format only works in version 1.43 or newer
# this example is intended for matching closely what previous versions rendered
logFormat: "%clientHost %l %user [%t{dd/MMM/yyyy:HH:mm:ss Z,UTC}] \"%requestURL\" %statusCode %bytesSent
%elapsedTime %elapsedTime"
```

Sample Data

In order to demonstrate IQ Server concepts and provide a safe 'sandbox' to learn the concepts, the IQ Server is configured to populate sample data for new installs. The sample data consists of an organization with an application underneath it.

⚠ We recommend you delete the sample data when it's no longer needed (i.e. you have onboarded your own organizations and applications).

If you do not want the sample data installed, find the existing createSampleData property in the config.yml and change it prior to running IQ server for the first time:

```
createSampleData: false
```

Advanced Server Configuration Using Java System Properties

In unusual cases where the config.yml cannot be modified easily, or you wish to temporarily override a value within it, an advanced option allows overriding values within it by passing special Java system properties when starting IQ Server.

Property overrides must start with prefix **dw.** - followed by a dot-separated path to the configuration value being overridden.

Example Java Command Overriding HTTP Proxy Settings with System Properties

```
java -Ddw.proxy.hostname=proxy.example.com -Ddw.proxy.port=8888 -jar nexus-iq-server-*.jar server
config.yml
```

Example Java Command Overriding baseUrl with System Properties

```
java -Ddw.baseUrl=http://iq.example.com -jar nexus-iq-server-*.jar server config.yml
```

Updating your Nexus IQ Server Configuration

Nexus IQ Server 1.43 and newer uses a more powerful configuration format for its networking and logging.

Due to this there have been some significant changes in how to write your Nexus IQ Server configuration file. This is a YAML formatted file called **`config.yml`** and can be found in the installation directory.

The main changes are as follows:

- The `http` section is replaced by the `server` section
- Application and admin `server` settings are separated under `applicationConnectors` and `adminConnectors`
- `adminUsername` and `adminPassword` are removed
- `connectorType` is replaced by `- type` which can be set to one of `http/https`
- `ssl` is replaced by using `- type: https` for an application/admin connector and putting the settings underneath
- Each logging target (e.g. `console`, `file`, etc) is specified under `appenders` as a `- type`
- Logging targets no longer have an `enabled` property (if they are specified then they are enabled)
- request.log line format has changed and is now configurable

Examples showing these changes are shown below:

Nexus IQ Server version 1.42 and lower

```
# Nexus IQ Server specific settings (not shown)
http:
  port: 8070
  adminPort: 8071
  adminUsername: user1234
  adminPassword: pass5678
  ssl:
    keyStore: /path/to/your/keystore/file
    keyStorePassword: yourpassword
  requestLog:
    console:
      enabled: false
    file:
      enabled: true
      currentLogFilename: ./log/request.log
      archivedLogFilenamePattern: ./log/request-%d.log.gz
      archivedFileCount: 50
logging:
  level: DEBUG
  console:
    enabled: true
    threshold: INFO
  file:
    enabled: true
    threshold: ALL
    currentLogFilename: ./log/clm-server.log
    archivedLogFilenamePattern: ./log/clm-server-%d.log.gz
    archivedFileCount: 50
```

Nexus IQ Server version 1.43 and later

```
# Nexus IQ Server specific settings (not shown) remain unchanged
# adminUsername and adminPassword no longer exist
# requestLog appenders logFormat must be explicitly added to maintain the previous versions default format
server:
  applicationConnectors:
    - type: https
      port: 8070
      keyStorePath: /path/to/your/keystore/file
      keyStorePassword: yourpassword
  adminConnectors:
    - type: https
      port: 8071
      keyStorePath: /path/to/your/keystore/file
      keyStorePassword: yourpassword
  requestLog:
    appenders:
      # No - type: console means it is disabled
```

```
      - type: file
        # preferred format as of version 1.43
        logFormat: "%clientHost %l %user [%date] \"%requestURL\" %statusCode %bytesSent %elapsedTime
\"%header{User-Agent}\""
        currentLogFilename: ./log/request.log
        archivedLogFilenamePattern: ./log/request-%d.log.gz
        archivedFileCount: 50
logging:
  level: DEBUG
  appenders:
    - type: console
      threshold: INFO
    - type: file
      threshold: ALL
      currentLogFilename: ./log/clm-server.log
      archivedLogFilenamePattern: ./log/clm-server-%d.log.gz
      archivedFileCount: 50
```

Note that these examples are heavily truncated for easy viewing and do not include any Nexus IQ Server specific configuration settings (see page 96), which remain unchanged.

If you previously used the `adminUsername` and `adminPassword` options, we recommend using a reverse proxy instead to shield and authenticate access to the operational services.

For more information on the new configuration format, consult the respective sections of the IQ Server Configuration:

- Logging Configuration (see page 0)
- HTTP Configuration (see page 99)
- HTTPS Configuration (see page 100)

IQ Server Administration

Topics in this section cover the creation and management of user accounts, passwords, and associated permissions.

> The topics in this section are aimed at system administrators. Most areas discussed here will require a user account that has been assigned to the System and/or Policy Administrator role.

Related topics

- Logging In (see page 110)
- User Management (see page 111)
- LDAP Integration (see page 113)
- Role Management (see page 121)

Logging In

At a minimum, IQ Server requires you to log in before anything else can happen. This can be done by creating a user within the IQ Server realm, or by configuring LDAP (see page 113), and logging in via one of those connected users.

To log into the IQ Server simply go to the address of your IQ Server (e.g. localhost:8070) and enter your username and password.

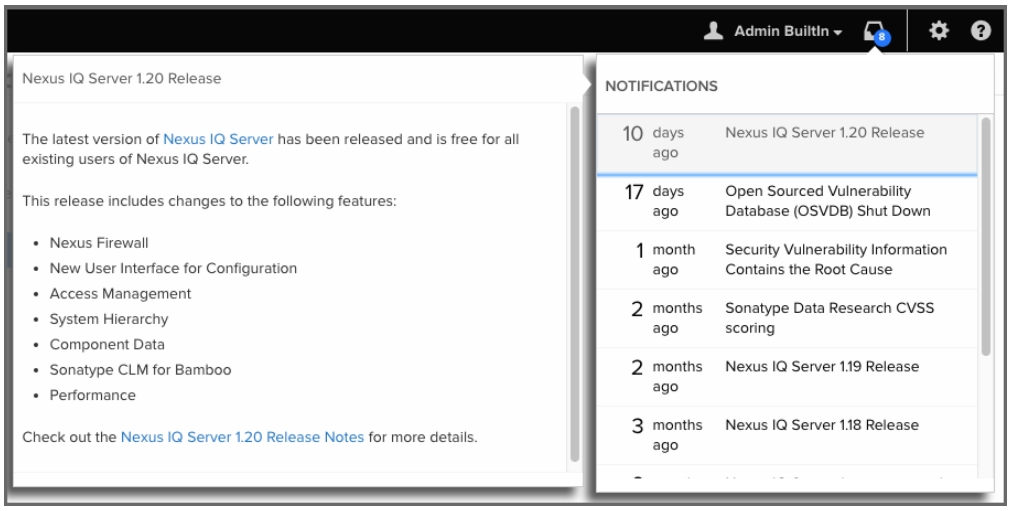
If this is your very first time logging in, you will need to use the default Admin account. This user is a preconfigured Admin account that has been assigned to all Administrator roles. Once you log in with this account for the first time, be sure to change the admin password.

To logout, click on the *Log Out* link located in the upper right corner.

> ⚠ IQ server will timeout after 30 minutes of inactivity.

Product Notifications

Once logged in, you can check for product notifications, which provide the most up-to-date information about IQ Server. Click the notifications icon on the IQ Server toolbar to view the Notifications panel. If you have unread notifications, they are indicated by a count, in blue, displayed over the notification icon.



User Management

Changing the Admin Account Password

The IQ Server ships with a default Admin account with a username admin and a password admin123. If you do nothing else related to security, be sure to change this password.

1. After logging in with the Admin account, click on the button with your username to the left of the System Preferences gear-shaped, icon. For the default administrator the username will show Admin Builtin.
2. A list of options will be displayed, click *Change Password*.
3. Enter the current password ("admin123" for the default admin user), the new password, and then confirm the new password.
4. Click the *Change* button to save the new password.

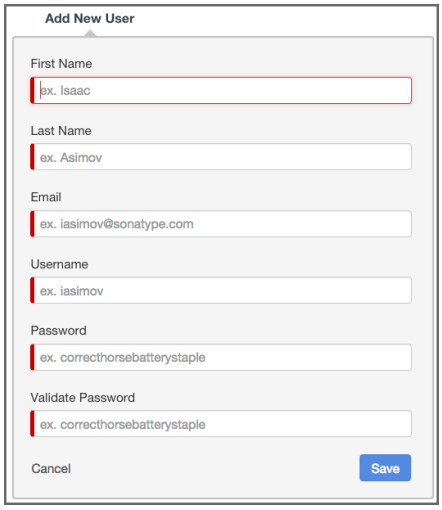
⚠ Any user, including an admin, can change their password following the instructions above. However, only an admin can reset a user's password without knowledge of the current password.

Creating a User

While we recommend using a security protocol such as LDAP for managing users and permissions, the IQ Server realm is still available for those who would like a lighter setup, where all users, groups and rights are stored directly in the IQ Server.

To create a new user in the IQ Server realm, follow the instructions below.

1. Log into the IQ Server with a user that has been assigned to the System Administrator role.
2. Click the System Preferences icon  located in the top right of the header.
3. Choose *Users* from the drop down menu. The Users administration area will now be displayed.
4. Click the *New User* button located at the top of the list of users.
5. The *Add New User* form will now be displayed. Enter the following information:
    a. First Name
    b. Last Name
    c. Email
    d. Username
    e. Password
    f. Validate Password
6. Click the *Save* button, to save the new user.

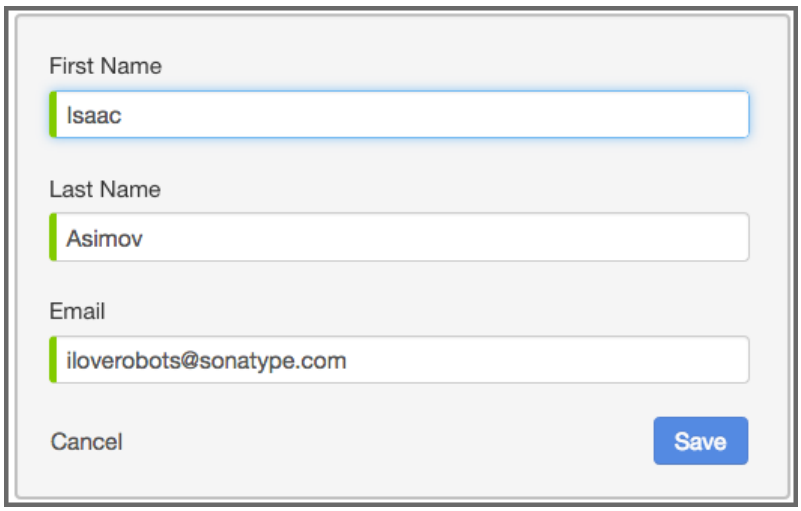


Editing and Deleting User Information

Editing user information is only available to an admin. The information that can be edited includes the first name, last name, email address, and password. To edit an existing user, follow these steps:

1. Log into the IQ Server with a user that has been assigned to the System Administrator role.
2. Click the System Preferences icon located in the top right of the IQ Server header.
3. Choose *Users* from the drop down menu. The Users administration area will now be displayed.
4. At least one user - the initial admin account - will be displayed. If you hover your pointer over the user record you will notice that there are three icons on the right.
    a. The icon shaped like a pencil will allow you to edit user information (i.e. first name, last name, and e-mail address).
    b. The icon shaped like a bag with an arrow back is for resetting a user's password. If you use this option a random, secure password will be generated and displayed in a dialog. Click the icon to the right of the field to copy it to clipboard.
    c. The icon shaped like a trashcan will allow you to delete the user after you confirm the deletion in a dialog.
5. Make any desired changes, and unless you chose to delete the record, click the *Save* button.

With regard to changing a user's password, a user can always change their own password. However, this requires knowledge of the existing password. If you encounter a user that has forgotten their password, you can reset it for them.



LDAP Integration

⚠ This section assumes you are familiar with LDAP (Lightweight Directory Access Protocol), and have an LDAP server currently in use.

You can configure IQ Server to work with an LDAP server for the purposes of authenticating users and managing users and groups.

Configuring LDAP Server Connection

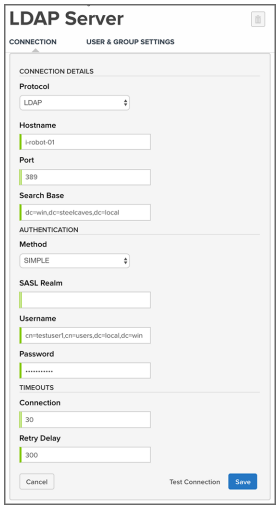
The first step in integrating IQ Server and LDAP is to configure the LDAP server connection as follows:

1. Log into the IQ Server using a user account assigned to the System Administrator role.
2. In the System Preferences menu ⚙ on the IQ Server toolbar, click *LDAP*.
3. In the LDAP Servers view, click *Add a Server*.
4. In the Enter Server Name box, type a name to identify the LDAP server. Click *Save*.
   The LDAP editor is displayed.
5. Enter the following parameters using values specific to your LDAP server:

Protocol	Valid values in this drop-down are LDAP and LDAPS, which correspond to the Lightweight Directory Access Protocol and the Lightweight Directory Access Protocol over SSL.
Hostname	The hostname or IP address of the LDAP server.

Port	The port on which the LDAP server is listening. Port 389 is the default port for the LDAP protocol and port 636 is the default port for the LDAPS protocol.
Search Base	The search base is the Distinguished Name (DN) to be appended to the LDAP query. The search base usually corresponds to the domain name of an organization. For example, the search base on the Sonatype LDAP server could be "dc=sonatype,dc=com".
Method	Four distinct authentication methods can be used when connecting to the LDAP Server: a. *NONE (Anonymous Authentication):* Used when you only need read-only access to non-protected entries and attributes when binding to the LDAP server. b. *SIMPLE*: Simple authentication is not recommended for production deployments not using the secure LDAPS protocol as it sends a clear-text password over the network. c. *DIGESTMD5*: This is an improvement on the CRAM-MD5 authentication method. For more information, see http://www.ietf.org/rfc/rfc2831.txt. d. *CRAMMD5*: The Challenge-Response Authentication Method (CRAM) based on the HMAC-MD5 MAC algorithm. In this authentication method, the server sends a challenge string to the client, the client responds with a username followed by a Hex digest which the server compares to an expected value. For more information, see RFC 2195. For a full discussion of LDAP authentication approaches, see http://www.ietf.org/rfc/rfc2829.txt and http://www.ietf.org/rfc/rfc2251.txt.
SASL Realm	The Simple Authentication and Security Layer (SASL) Realm to connect with. Not available if authentication method is NONE.
Username	Username of the Administrative LDAP User to connect (or bind) with. This is a Distinguished Name of a user who has read access to all users and groups.
Password	Password for an Administrative LDAP User.
Connection	The number of seconds to try and connect to the configured server before returning an error.
Retry Delay	The number of seconds to wait before attempting to connect to the configured server again (after an error).

6. Click **Test Connection** to verify a connection can be made to the LDAP server.
7. Click **Save** when finished.



Mapping LDAP Users

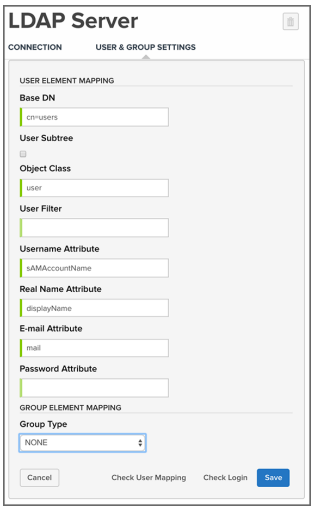
After configuring an LDAP server connection, the next step in integrating IQ Server and LDAP is to map LDAP users as follows:

1. In the LDAP editor, click the *User & Group Settings* tab.
2. Enter the following parameters under User Element Mapping using values specific to your LDAP server:

Base DN	*(required)* Corresponds to the Base DN (Distinguished Name) containing user entries. This DN is going to be relative to the Search Base. For example, if your users are all contained in "cn=users,dc=sonatype,dc=com" and you specified a Search Base of "dc=sonatype,dc=com" you would use a value of "cn=users"
User Subtree	Enable this parameter if there is a tree below the Base DN which can contain user entries. For example, if all users are in "cn=users" this field should not be toggled. However, if users can appear in organizational units below "cn=users", such as "ou=development,cn=users,dc=sonatype,dc=com" this field should be toggled
Object Class	*(required)* The object class defines what attributes are expected for a given object. What is entered here must be the object class for the Username Attribute, Real Name Attribute, Email Attribute, and the Password Attribute.
User Filter	The user filter allows you to isolate a specific set of users under the Base DN.
Username Attribute	*(required)* This is the attribute of the Object class which supplies the username.

Real Name Attribute	*(required)* This is the attribute of the Object class which supplies the real name of the user.
E-mail Attribute	*(required)* This is the attribute of the Object class which supplies the email address of the user.
Password Attribute	This is the attribute of the Object class which supplies the User Password. By default it is not required, which means authentication will occur as a bind to the LDAP server. Otherwise this is the attribute of the Object class which supplies the password of the user.

3. Click *Save* when finished.



Mapping LDAP Groups

In most LDAP implementations, users are mapped into groups with different permissions. If LDAP groups are not mapped, then all LDAP users are pulled in from the Base DN, which may give unintended access to some users.

After configuring an LDAP server[95] and mapping LDAP users[96], the last step in integrating IQ Server and LDAP is to map the LDAP groups as follows:

1. In the LDAP editor, click the *User & Group Settings* tab.
2. In the Group Element Mapping section, select a *Group Type*.
    a. For a Dynamic group, enter a *Member of Attribute* and use the slider to enable or disable *Group Search*.

95 https://help.sonatype.com/iqserver/iq-server-administration/ldap-integration#LDAPIntegration-ConfiguringLDAPServerConnection
96 https://help.sonatype.com/iqserver/iq-server-administration/ldap-integration#LDAPIntegration-MappingLDAPUsers

b. For a Static group, enter the following parameters using values specific to your LDAP server: *Base DN, Object Class, Group ID Attribute, Group Member Attribute,* and *Group Member Format.*

3. Click *Save* when finished.

Disabling group search for Dynamic Groups may improve performance but will not return group results when searching IQ Server users.

LDAP Group Parameters

Groups are generally one of two types in LDAP systems: static or dynamic. For static groups, users are explicitly assigned to the group. For dynamic groups, users are assigned to the group based on a set of common attributes.

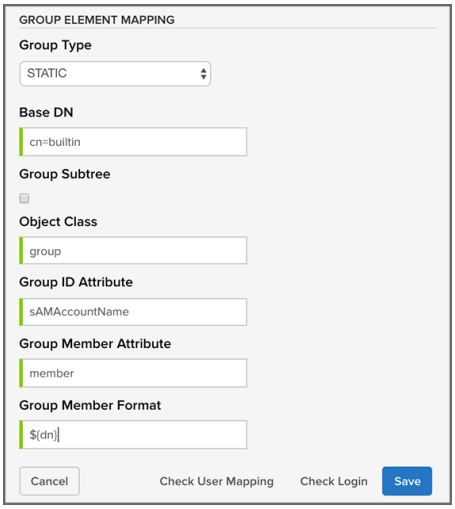
Static groups are preferred over dynamic ones, and will generally perform better if you have a large number of LDAP users.

When you map LDAP groups, some parameters are required and some are optional. The required parameters are noted below.

Static Groups

Static groups are configured with the following parameters:

Base DN	*(required)* This field is similar to the Base DN field described for User Element Mapping. If your groups were defined under "ou=groups,dc=sonatype,dc=com", this field would have a value of "ou=groups"
Group Subtree	This field is similar to the User Subtree field described for User Element Mapping. If all groups are defined under the entry defined in Base DN, this field should not be selected. If a group can be defined in a tree of organizational units under the Base DN, this field should be selected.
Object Class	*(required)* This is a standard object class defined as a collection of references to unique entries in an LDAP directory, and can be used to associate user entries with a group.
Group ID Attribute	*(required)* This field specifies the attribute of the Object class that defines the Group ID.
Group Member Attribute	*(required)* This field specifies the attribute of the Object class that defines a member of a group.
Group Member Format	*(required)* This field captures the format of the Group Member Attribute, and is used to extract a username from this attribute. For example, if the Group Member Attribute has the format uid=brian,ou=users,dc=sonatype,dc=com, then the Group Member Format would be uid=${username},ou=users,dc=sonatype,dc=com. If the Group Member Attribute had the format "brian", then the Group Member Format would be ${username}.

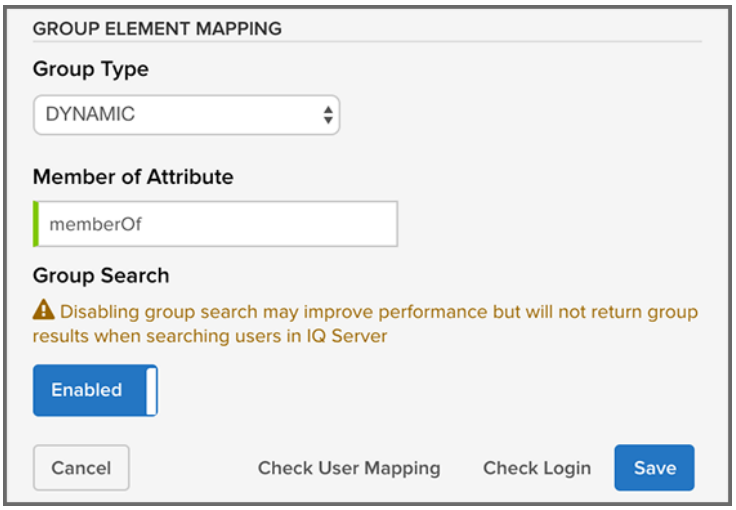


Dynamic Groups

If your installation does not use static groups, you can configure LDAP integration to refer to an attribute on the User entry to derive group membership. To do this, select *Dynamic Groups* in the *Group Type* field.

Member of Attribute (required): Dynamic groups are configured via the Member of Attribute parameter. This attribute of the user entry will provide a list of LDAP groups that the user is a member of. In this configuration, a user entry would have an attribute such as memberOf which would contain the name of a group.
Group Search: Depending on the size of your enterprise, LDAP search could be slow. If you find this is the case, click the Enabled slider below Group Search to disable group search.

Disabling group search will exclude groups from search results when assigning users to roles. Searching for users will remain unaffected.

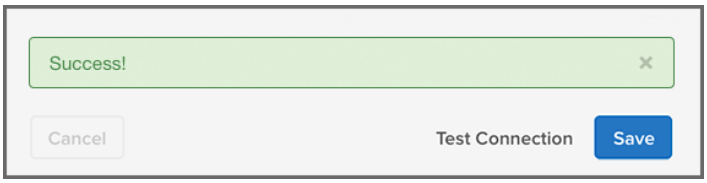


Verifying LDAP Configuration

The LDAP editor provides several ways to verify your LDAP configuration in IQ Server.

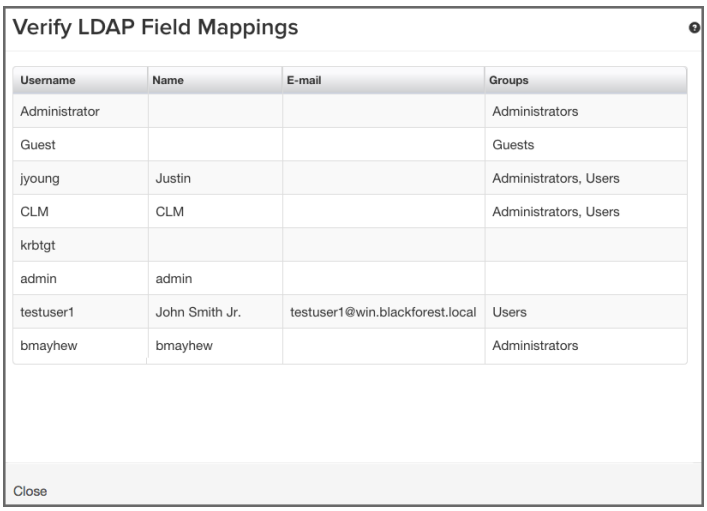
Test Connection

Click **Test Connection** to verify your LDAP configuration. If you can't connect to your LDAP server, user and group mapping will fail as well.



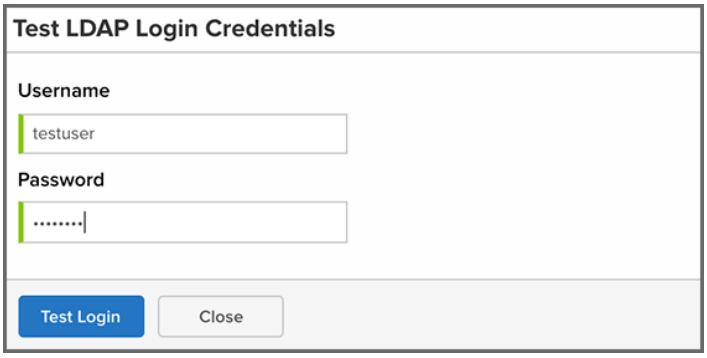
Check User and Group Mapping

Click Check User Mapping to verify that usernames, real names, email addresses, and groups have been mapped correctly.



Check Login

To ensure users can log in, click *Check Login* to open the "Test LDAP Login Credentials" dialog box. Enter a username and password, and click *Test Login*.



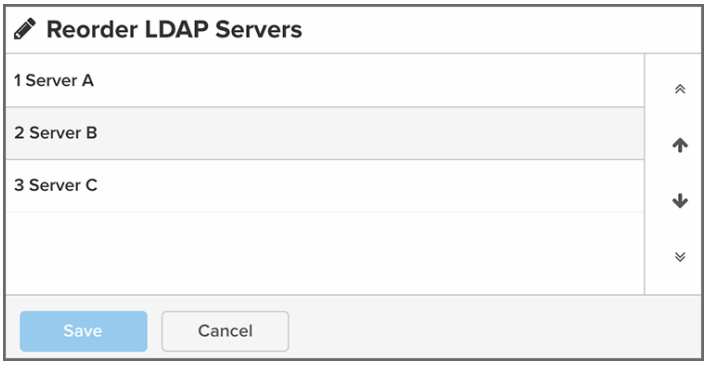
Reordering LDAP Servers

⚠ Reordering LDAP servers may impact user authentication and/or authorization.

When authenticating a user, IQ Server connects to LDAP servers in the order displayed in the LDAP Servers view. To manage the order, perform the following steps:

1. In the LDAP Servers view, click *Reorder List*. The *Reorder LDAP Servers* modal opens.
2. Click on a server and use the up and down arrow buttons to reorder the list. Double arrow buttons move the selection to the top or bottom of the list.
3. Click *Save* when you are finished.

The server list in the LDAP Servers view updates to reflect the new order.



Role Management

Overview

Roles provide a set of permissions that grant various levels of access and control over the IQ Server as well as the connected suite of tools. To grant permissions, you assign a user to either a system-wide administrator role or an organizational role at one of the levels in the system hierarchy: root organization, organization, or application. Which role and level you choose for a user determines what permissions that user receives.

You can assign roles to individual users or groups of users. IQ Server has a built-in group called Authenticated Users that contains any authenticated user. In addition, LDAP groups may be available, if you configured IQ Server to use an LDAP server with users mapped to specific groups.

Role Hierarchy

The scope of permissions granted to a role is governed by where that role is assigned in the system hierarchy. A role assigned to:

- Root organization - Grants permissions to all organizations, applications, and repositories.
- Organization - Grants permissions to that individual organization and any applications attached to it.
- Application - Grants permissions only to the individual application.

Firewall solution users have an additional entity for which roles can be assigned:

- Repositories - Grants permissions to repositories.

Built-in Roles

IQ Server has several built-in roles, which are shown below. If one does not suit your needs, you can create a custom role.

Administrator Roles:

- System Administrator - Manages system configuration and users, which includes LDAP and product license management as well as the ability to assign other users to the System Administrator role.
- Policy Administrator - Provides full control over organizations, applications, policies, policy violations and custom roles. Only the Policy Administrator has the ability to create organizations.

Organizational Roles:

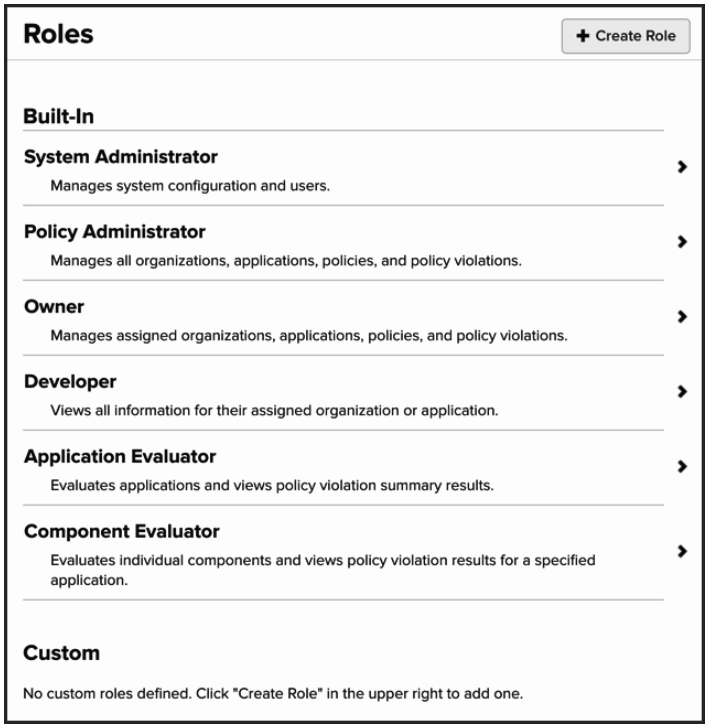
- Owner - Manages assigned organizations, applications, policies, and policy violations.
- Developer - Views all information for their assigned organization or application.

- Application Evaluator - Evaluates applications and views policy violation summary results.
- Component Evaluator - Evaluates individual components and views policy violation results for a specified application.

To view roles in IQ Server:

1. Click the *System Preferences* icon on the IQ Server toolbar.
2. Click *Roles* on the System Preferences submenu. A list of built-in roles is displayed.

> Only a user assigned to an administrator role can see the information below. If you are using the built-in Admin user account, it is assigned to all administrator roles. It is highly recommended that you change the Admin password.



Built-in Role Permissions

To view permissions assigned to built-in roles:

1. Click the *System Preferences* icon on the IQ Server toolbar.
2. Click *Roles* on the System Preferences submenu. A list of roles is displayed.
3. Click the arrow next to a specific role to view its details and permissions.
4. The built-in roles have the permissions shown below.

Administrator Roles

Permissions	System Administrator	Policy Administrator
Administrator Permissions		
Edit System Configuration and Users	👍	⊘
Edit Custom Roles	⊘	👍
View All Roles	👍	👍
IQ Permissions		
Edit Proprietary Components	⊘	👍
Claim Components	⊘	👍
Edit IQ Elements	⊘	👍
View IQ Elements	⊘	👍
Evaluate Applications	⊘	👍
Evaluate Individual Components	⊘	👍
Add Applications	⊘	👍
Manage Automatic Application Creation	⊘	👍

Organizational Roles

Permissions	Owner	Developer	Application Evaluator	Component Evaluator
Administrator Permissions				
Edit System Configuration and Users	⊘	⊘	⊘	⊘
Edit Custom Roles	⊘	⊘	⊘	⊘

View All Roles	👍	🚫	🚫	🚫
IQ Permissions				
Edit Proprietary Components	👍	🚫	🚫	🚫
Claim Components	🚫	🚫	🚫	🚫
Edit IQ Elements	👍	🚫	🚫	🚫
View IQ Elements	👍	👍	🚫	🚫
Evaluate Applications	👍	🚫	👍	🚫
Evaluate Individual Components	👍	👍	🚫	👍
Add Applications	👍	🚫	🚫	🚫
Manage Automatic Application Creation	🚫	🚫	🚫	🚫

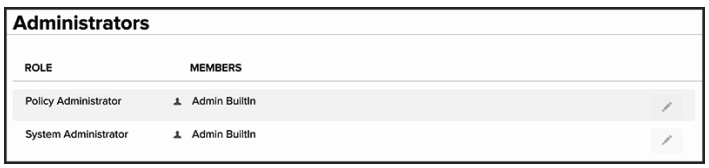
⚠ IQ Elements includes organizations, applications, policies, component labels, license threat groups, application categories, policy violations and waivers.

Managing Administrator Roles

To manage administrator roles, you must log into IQ Server as a user assigned to the System Administrator role. By default, the built-in Admin user account is assigned to the System Administrator role.

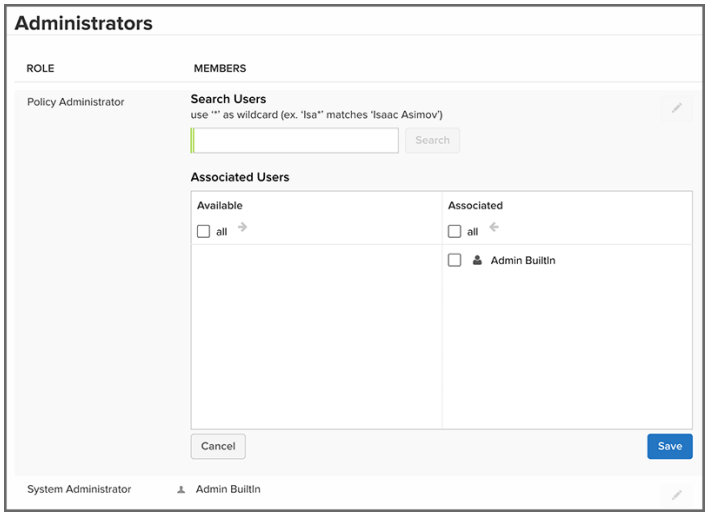
Viewing Administrator Roles

1. Click the System Preferences icon ⚙ on the IQ Server toolbar.
2. Click *Administrators*. A list of administrator roles and assigned members is displayed.



Assigning Users to Administrator Roles

1. Click the System Preferences icon on the IQ Server toolbar.
2. Click *Administrators*. The Administrators view is displayed.
3. Click the *Edit Role* button (looks like a pencil) for the role to which you want to add users. The Administrators view is updated to display the role and its members
4. Search for a user you want to add to the role by entering a full name or part of a name with an asterisk into the search box, then click Search. You can use an asterisk as a wildcard at the beginning or end of a character string. For example, isa*, *mov, and *asi* will all match the name, "Isaac Asimov." Any matching names are displayed in the Available list.
5. To add a user to the role, click a user's name in the Available list and use the right arrow to move the name to the Associated list. To remove a user from a role, click a user's name in the Associated list and use the left arrow to move the name to the Available list.
6. Click *Save* to save the role assignment(s).



Managing Organizational Roles

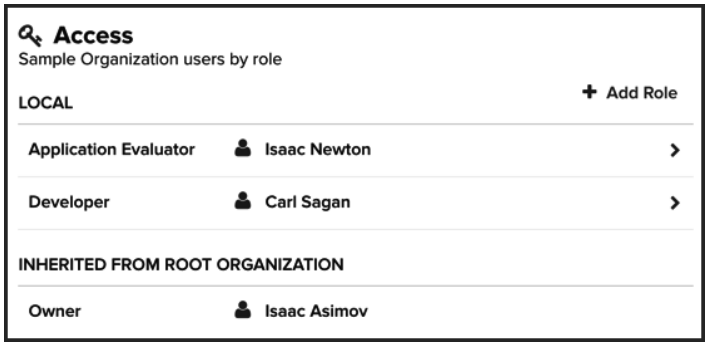
To manage organizational roles, you must log into IQ Server as a user assigned to the Policy Administrator role or Owner role. By default, the built-in Admin user account is assigned to the Policy Administrator role.

Viewing Organizational Role Assignments

To view organizational role assignments:

1. Click the Organization & Policies icon on the IQ Server toolbar.
2. Select an entity (organization, application, or Repositories) in the sidebar.

3. Click *Access* in the menu bar at the top of the page to scroll to the Access section. Assigned roles are grouped as follows:
    a. Local - Role assignments with a scope that's specific to the selected organization or application.
    b. Inherited - Role assignments derived from an organization that's higher in the system hierarchy than the currently selected organization or application.

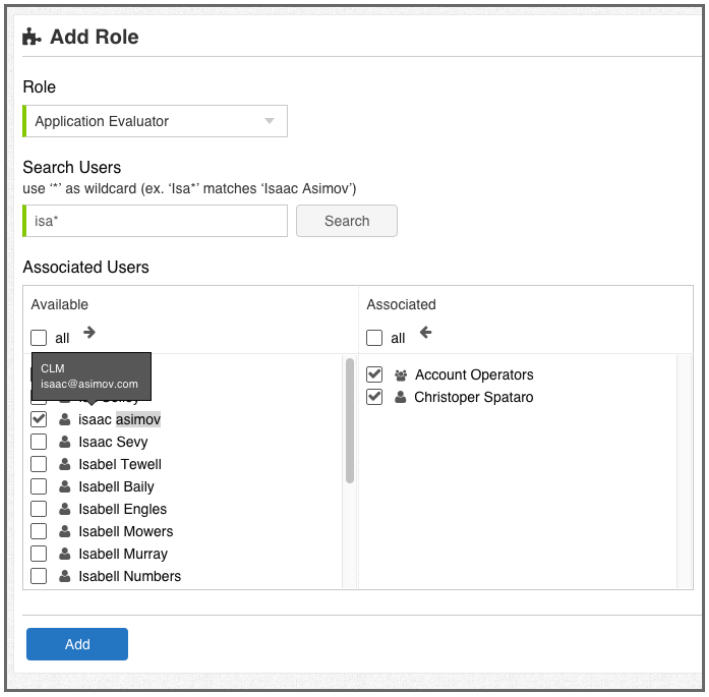


Assigning Users to Organizational Roles

To assign a user to an organizational role:

1. Click the Organization & Policies icon on the IQ Server toolbar.
2. Select an entity (organization, application, or Repositories) in the sidebar.
3. Click *Access* in the menu bar at the top of the page to scroll to the Access section.
4. Click the *Add Role* button. The Access editor is displayed.
5. In the *Role* box, select a user role.
6. In the *Search Users* box, search for a user by entering a full name or part of name with an asterisk, then click **Search**. You can use an asterisk as a wildcard at the beginning or end of a character string. For example, isa*, *mov, and *asi* will all match the name, "Isaac Asimov." Any matching names are displayed below in the Associated Users list.
7. In the *Associated Users* list, select a user in the *Available* column on the left, then click the right arrow button to move the user to the Associated column on the right. If you accidentally add a wrong user, select the user in the Associated column, then click the left arrow to return the user to the Available column.
8. Click *Add*.

⚠ If you integrated an LDAP server with IQ Server, the LDAP users and groups are also displayed in the search results. If you hover over a list item, the LDAP realm and email address are displayed when available.



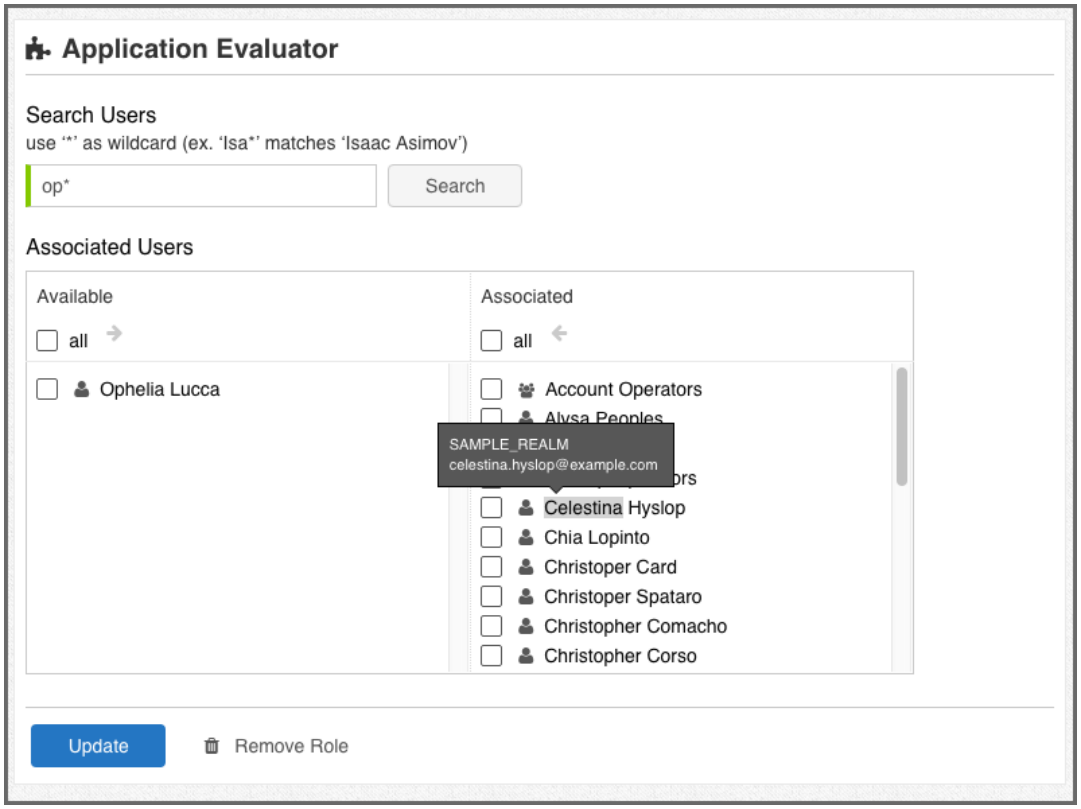
If you want to continue adding role assignments for the selected organization, application or Repositories, click Add Role in the sidebar.

Editing Organizational Role Assignments

To edit an organizational role assignment:

1. Click the Organization & Policies icon  on the IQ Server toolbar.
2. In the sidebar, select the entity (organization, application or Repositories) in which the role is assigned.
3. Click *Access* in the menu bar at the top of the page to scroll to the Access section.
4. Click a listed role (or the chevron next to its name) to display the *Access* editor.
5. In the *Search Users* box, search for a user by entering a full name or part of name with an asterisk, then click **Search**. You can use an asterisk as a wildcard at the beginning or end of a character string. For example, isa*, *mov, and *asi* will all match the name, "Isaac Asimov." Any matching names are displayed below in the *Associated Users* list.
6. In the *Associated Users* list, you can add a user to a role by selecting the user in the Available column on the left and clicking the right arrow button to move the user to the Associated column. To remove a user from a role, select the user in the Associated column, then click the left arrow to return the user to the Available column.
7. Click **Update**.

⚠ If you integrated an LDAP server with IQ Server, the LDAP users and groups are also displayed in the search results. If you hover over a list item, the LDAP realm and email address are displayed when available.



✓ If you want to continue adding role assignments for the selected organization, application or Repositories, click Add Role in the sidebar.

Removing Organizational Role Assignments

To remove organizational role assignments:

1. Click the Organization & Policies icon on the IQ Server toolbar.
2. In the sidebar, select the entity (organization, application or Repositories) in which the role is assigned.
3. Click *Access* in the menu bar at the top of the page to scroll to the Access section.
4. Click a listed role (or the chevron next to its name) to display the *Access* editor.

5. Click the *Remove Role* button, then click *Continue* to remove the role or click *Cancel* to keep the role.

⚠ This is the equivalent of removing or disassociating all users from a role.

Custom Roles

You must have permission to Edit Custom Roles if you want create a custom role. The default Admin account and the built-in Policy Administrator role have this permission.

Custom roles allow you to fine tune IQ security permissions for different users. The following permissions are available for custom roles:

Administrator Roles:

- View All Roles
- Edit Proprietary Components

Organizational Roles:

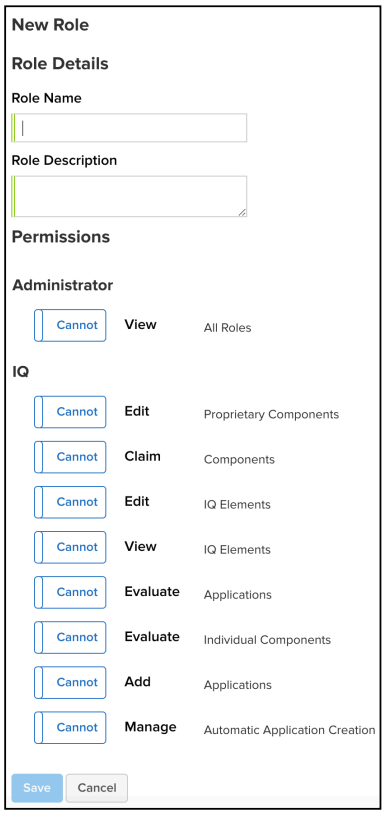
- Claim Components
- Edit IQ Elements
- View IQ Elements
- Evaluate Applications
- Evaluate Individual Components
- Add Applications
- Manage Automatic Application Creation

To achieve desired behavior in the IQ user interface, you may need to assign View IQ Elements along with other permissions. For example, to allow a user to create applications in an organization but not edit the organization, you should add View IQ Elements and Add Applications to the role.oh sh

To create a custom role:

1. Click the System Preferences icon  on the IQ Server toolbar and then click *Roles*.
2. Click the *Create Role* button.
3. Enter a name and description for the role.
4. Click the *Can/Cannot* slider to enable or disable a permission as desired.
5. Click the *Save* button.

⚠ Whenever a user assigned to a custom role with Add Applications permission creates an application, that user is automatically assigned to the Owner role for that application.



Assigning Groups to Roles without Searching

If you have an LDAP configuration that prohibits searching for groups, then the Access editor will have an additional section called Associate Group. You can use this section to enter manually a group name and add it to a role.

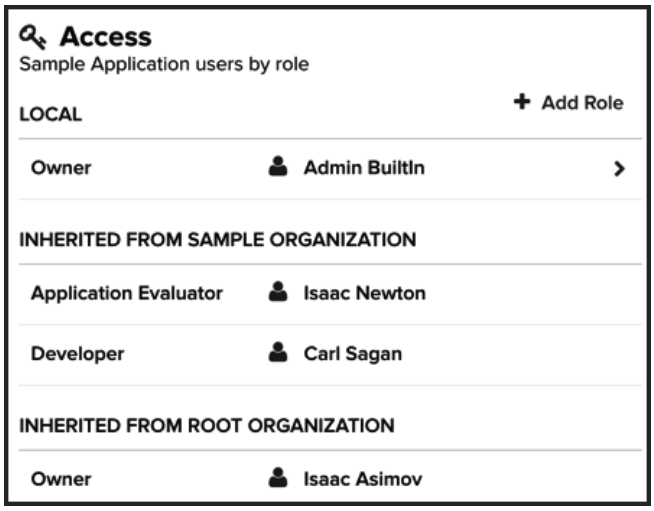
To assign groups to roles without searching:

1. Click the Organization & Policies icon  on the IQ Server toolbar.
2. In the sidebar, select an entity (organization, application or Repositories).
3. Click *Access* in the menu bar at the top of the page to scroll to the *Access* section.
4. Open the Access editor by clicking *Add Role* or the chevron next to an existing role.
5. In the *Associate Group* box, enter the group name. The text must be an exact match.
6. Click *Add*. The group name is added to the *Associated* column without performing a search of users and groups.

Role Assignments

To view role assignments:

1. Click the Organization & Policies icon on the IQ Server toolbar.
2. Select an organization or application in the sidebar. A page of customizable settings is displayed.
3. Click *Access* in the menu bar at the top of the page to scroll to the Access section. Users are displayed by their assigned roles for the selected entity (i.e. organization or application). The information is grouped by where the role assignments were made: locally in the current entity or inherited from an entity higher in the system hierarchy.



Organization and Application Management

When you launch the IQ Server for the first time, even after setting up and configuring your security parameters, there will be little to no information displayed. In many ways, it will look like a blank slate.

Before you can go further with anything else (like evaluating your applications and policy management), you must create at least one organization and one application.

In this section, we cover details around managing organizations and applications. We'll start off with some basics around the relationship, and then get right into how to create them.

Related topics

- Hierarchy and Inheritance (see page 132)
- Root Organization (see page 133)
- Application Management (see page 136)
- Organization Management (see page 142)

- Managing Automatic Applications (see page 144)

Hierarchy and Inheritance

Hierarchy

While we'll cover policy management a little bit later, it's important to understand policy management is centered around a principal of hierarchy. Specifically, a root organization is at the top of the hierarchy. Below it fall organizations, and then applications.

For many teams, the structure of organizations and applications will follow their own "command and control". In this instance, various business units are responsible for top level policies, and then they may add policies that are specific to their organization, or even to specific applications. As such, each business unit will have its own organization below the root organization, and those organizations will have unique applications below them.

For others, the applications their teams work on create more logical categories, such as "Internal". This allows a business to mimic commercial units where each business unit is a separate organization with applications below it.

For now though, the important pieces to remember are:

- The root organization acts as a container for organizations.
- Those organizations act as containers for applications.
- The root organization has no applications attached to it.
- Applications can only be attached to a single organization.

In the next section we'll talk more about the flow through these various pieces, which we refer to as inheritance.

Inheritance

Every organization has rules around component usage. While we will discuss policy in much greater detail in the Policy Management chapter, for now, let's just consider policy as a representation of all the rules and actions your business has to help identify which components your development teams should be using.

With rules as guidance, your teams will then simply follow that advice and go no further, or they might develop additional rules given very specific needs. Whatever the case, all teams are still accountable for a core set of rules.

Now, let's think about how those move through our organization. We like to refer to this flow of rules, or policy, as inheritance. In general, whatever is specified for the root organization will extend out to all organizations. And, whatever is specified for an organization will extend out to any attached applications.

Of course there's opportunity for additional customization here as well. However, the real advantage to inheritance is when you want to make changes that affect more than one organization or one application. For example, if you managed everything at the application level, one change needed for all applications would need to be made to each one. That's not hard if you have ten applications, but what if you have a hundred, or thousands even? The same applies for organizations.

It is recommended that you use the root organization to set policy that applies globally to all organizations. You can fine tune policy at the organization level and allow your applications to inherit those changes.

Root Organization

⚠ The Root Organization was introduced in IQ Server version 1.18. Your installation of the IQ Server may be affected depending on its version:

- If your original installation is version 1.18 or later, then the Root Organization is already part of your IQ Server system hierarchy. You can skip this section of the documentation.
- If you upgraded from version 1.17 (or earlier) to version 1.18 (or later), then you need to create the Root Organization by following the instructions here.

The Root Organization is a new entity at the top of the system hierarchy that allows you to set policy globally across all organizations and applications. Creating the Root Organization is a one time process, and occurs when the IQ Server is restarted. The process makes a permanent change to the system hierarchy that cannot be undone. For this reason, it is strongly recommended that you backup the IQ Server before you create the Root Organization. For information on backing up the IQ Server, see Backing Up the IQ Server in the IQ Server Setup chapter.

Configuring the Root Organization

The next step in the creation process is deciding how to configure the Root Organization. You have two choices:

- Choose an existing organization to act as a template for the Root Organization, which is configured with all of the policies and policy elements that you want to move to the Root Organization.
- Choose to create an empty Root Organization to which you can later add policy. To help you decide between the two, each is described in more detail below.

Using a Template Organization

Using a template organization enables you to take the configuration of an existing organization and move it to the Root Organization. The policy elements to be moved include all policies, component labels, license threat groups, application categories, and policy monitoring, but not security settings.

When the Root Organization is created, the policy elements in the template organization are compared by name to policy elements throughout the IQ Server hierarchy. Only names are compared, not attributes like constraints, notifications, or applied licenses. When a match is found, all references to the that policy element are changed to refer to the template organization policy element and the match is deleted. The policy element is now inherited from the Root Organization rather than its original organization.

Using a template organization saves you time and effort in configuring the Root Organization, especially if you have many policy elements that you want to use globally throughout your system hierarchy. For this reason, it is the recommended method for configuring the Root Organization.

Creating an Empty Root Organization

If you choose to create an empty Root Organization, it will be blank like any other newly created organization; it will have no policies, component labels, license threat groups, or application categories. You can certainly add policy later to the Root Organization. However, if you want to move a policy element from an existing organization, then the migration process is a manual task. It involves renaming elements and switching between organizations multiple times, which can take time and effort.

Creating the Root Organization

Only users assigned to the Policy Administrator role can initiate the Root Organization creation process. Once the Root Organization is created, its security permissions behave the same as with any other organization. For more information about security, see the Security Administration chapter.

To Create the Root Organization:

1. Go to the *Root Organization* banner that appears below the IQ Server toolbar and click the *Get Started* button.
2. Select how you want to configure the Root Organization and click *Continue*. Choosing an existing organization to act as a template for the Root Organization configuration is strongly recommended.
3. Restart the IQ Server to complete the Root Organization creation process.

When the IQ Server is restarted, the Root Organization is created. The IQ Server hierarchy is permanently changed; the Root Organization is at the top, followed by organizations, then applications.

Viewing the Root Organization

When you launch the IQ Server for the first time, it has no organizations or applications except for the root organization. The root organization is always present and cannot be deleted. To view the root organization, click the Organization & Policies icon  on the IQ Server toolbar. The root organization appears in the sidebar.

> ⚠ You may also see Repositories in the sidebar of the Organization & Policies area. If you are a Nexus Repository Manager customer who has enabled the Audit and Quarantine features, you can use Repositories to set security for repository evaluation results.

The root organization allows you to set policy (rules) that are inherited globally by all organizations. This includes policy management features like policies, labels, license threat groups, application categories, and security, all of which are discussed in later chapters (see Basic Policy Management and Advanced Policy Management).

If you have permissions, you can change the name and icon attributes of the root organization:

1. Click the Organization & Policies icon  on the IQ Server toolbar.
2. Click the Root Organization icon  in the sidebar.
3. Go to the Actions menu and click *Edit Org Name / Icon*.
4. In the *Edit Organization* dialog, set the following attributes:
    a. Organization Name - Enter a name into the text box.
    b. Icon - Select from three options:
        i. Default - Uses the default image.
        ii. Upload a custom icon - Click the *Upload* Icon Image button to navigate to an image file in PNG format and click Open. The image should be sized to 160 x 160 pixels. Images with different sizes will be scaled.
        iii. Get a robot - Click the *Get Another Robot* button to cycle randomly through a variety of robot images.
5. Click the *Update* button.

Application Management

Overview

An application represents the link between what your team is actually developing and all the information that can be provided. You'll generally want this to be one-to-one. That is for every application your team is developing, you should create an application via the IQ Server. While this application is merely a representation of the real thing, it's no less important.

At the application level, you can do nearly everything you can for an organization. This includes creating policies and all the associated details. Additionally, whenever you perform an evaluation, and a report is generated, all that information will be associated to a specific application.

Creating an Application

As a general rule with any activity, make sure you have proper access to the IQ Server. In the case of application creation, you will need to be a member of a role that has been granted the *Edit IQ Elements* permission. In the roles provided by default, this would be the *Owner* role for an organization.

When you create an application, you can assign the following items for identifying it:

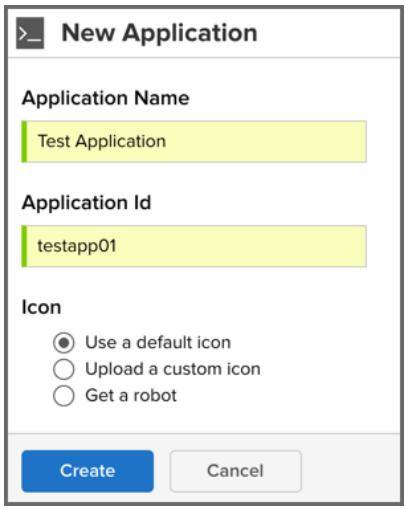
- *Application Name* - The name that appears in the sidebar and at the top of the Organization & Policies area when the application is selected.
- *Application ID* - A unique identifier that's used by integration tools (e.g. Nexus IQ for Bamboo, IQ for IDEA, REST APIs and more) when performing component evaluations.
- *Icon* - A graphical representation of the application that appears at the top of the Organization & Policies area (when the application is selected in the sidebar) and in Reporting. You can select from the default icon, a robot icon, or a custom icon.

To create an application:

1. Click the *Organization & Policies* icon  on the IQ Server toolbar.
2. In the sidebar, click the organization to which you want to attach an application. A New Application button is displayed.
3. Click the *New Application* button.
4. In the New Application dialog, set the following attributes:
    a. *Application Name* - Enter a name into the text box.
    b. *Application ID* - Enter an identifier into the text box.
    c. *Icon* - Select from three options:
        i. Default - Uses the default image.

ii. Upload a custom icon - Click the *Upload* icon image button to navigate to an image file in PNG format and click *Open*. The image should be sized to 160 x 160 pixels. Images with different sizes will be scaled.
iii. Get a robot - Click the *Get Another Robot* button to cycle randomly through a variety of robot images.

5. Click the *Create* button.



Editing an Application

1. Click the *Organization & Policies* icon  on the IQ Server toolbar.
2. In the sidebar, select the application you want to edit.
3. In the *Organization & Policies Management* area, go the *Actions* menu and click *Edit App Name / Icon*.
4. In the *Edit Application* dialog, edit any of the following attributes:
    a. *Application Name* - Enter a different name into the text box.
    b. *Icon* - Select from three options:
        i. Use existing icon - To use the current image.
        ii. Upload a custom icon - Click the *Upload* Icon Image button to navigate to an image file in PNG format and click *Open*. The image should be sized to 160 x 160 pixels. Images with different sizes will be scaled.
        iii. Get a robot - Click the *Get Another Robot* button to cycle randomly through a variety of robot images.

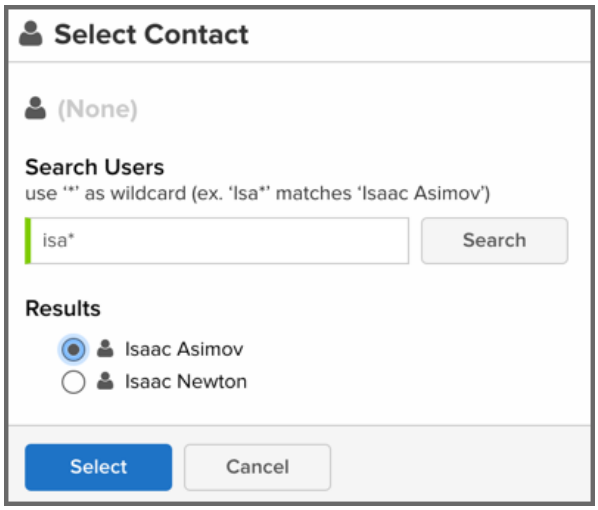
5. Click the *Update* button to save changes.

Selecting an Application Contact

Optionally, you can select a contact person for an application. The contact is displayed at the top of the *Organization & Policies* page, as well as in the reporting area of the Nexus and the PDF version of the report.

To select a contact:

1. Click the *Organization & Policies* icon on the IQ Server toolbar.
2. Select an application in the sidebar.
3. Go to the *Actions* menu and click *Select Contact*.
4. In the *Select Contact* box, search for a user by entering a full name or part of name with an asterisk, then click *Search*. You can use an asterisk as a wildcard at the beginning or end of a character string. For example, isa*, *mov, and *asi* will all match the name, "Isaac Asimov." Any matching names are displayed below the search box.
5. Select a user in the results list and click *Select*. The user's name is displayed at the top of the page below the application name.



Removing an Application Contact

To remove a contact:

1. Click the *Organization & Policies* icon on the IQ Server toolbar.
2. Select an application in the sidebar.
3. Go to the *Actions* menu and click *Select Contact*.
4. In the *Select Contact* box, click the *Clear Contact* button. In the alert box, click *Continue*. The user's name is removed from the top of the page.

Copying the Application ID to Clipboard

External tools integrated with IQ Server need to be configured with a public Application ID that identifies which application to use when evaluating components. You can speed the configuration process by copying the Application ID to the clipboard using the steps below.

To copy the Application ID to the clipboard:

1. Click the *Organization & Policies* icon  on the IQ Server toolbar.
2. Select an application in the sidebar.
3. Go to the *Actions* menu and click *App ID to Clipboard*.

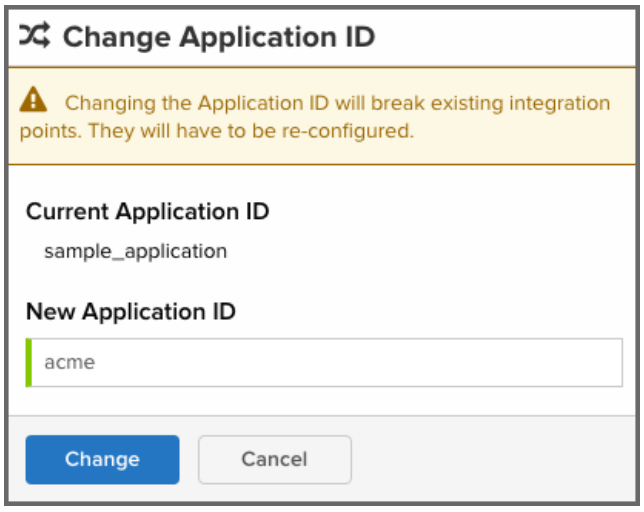
Changing an Application ID

The Application ID is a unique identifier used by external tools to integrate with IQ Server (e.g. Nexus IQ for Bamboo, IQ for IDEA, REST APIs, etc.) for performing component evaluations. If you change the Application ID, you must also reconfigure the external tool(s).

To change the Application ID for an application:

1. Click the *Organization & Policies* icon  on the IQ Server toolbar.Select an application in the sidebar.
2. Go to the *Actions* menu and click *Change App ID*.
3. In the *Change Application ID* dialog, enter a unique identifier in the *New Application ID* text box.
4. Click *Change* to save the new Application ID.

⚠ Remember to reconfigure your external tools to use the updated Application ID.



Moving an Application

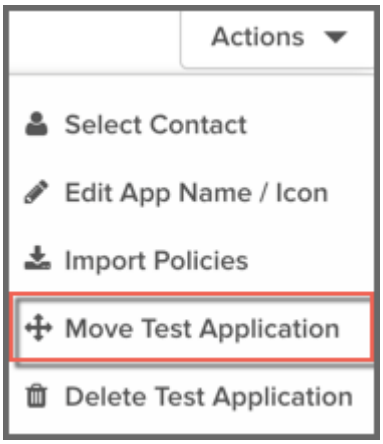
At some point, you may want to move an application to a different organization. When moving an application, it's not just the entity that moves, but also its configuration (i.e. policies, component labels, and waivers). If the application has any local configuration settings, those settings are unaffected and remain unchanged by a move. However, if the application has inherited any organization-level configuration, there is the potential for incompatibility between the application's source organization and the destination organization.

During a move, IQ Server compares the configuration of the source organization to the destination organization, and looks for matches by name. If the destination has configuration settings that match all of the source's configuration, then the application's parent reference is changed to the destination organization. If the source configuration is not matched in entirety, then IQ Server displays an error message with information about the missing configuration in the destination organization. You need to fix any discrepancy to ensure the source configuration is matched in the destination.

To move an application:

> Make sure you have permission to modify the application in its original parent organization and permission to create applications in the destination organization.

1. Click the *Organization & Policies* icon on the IQ Server toolbar.Select an application in the sidebar.
2. Select an application in the sidebar.
3. Go to the *Actions* menu and click *Move <application name>*.



4. In the *Move Application* dialog, select an organization from the *New Parent Organization* list and click *Update*.

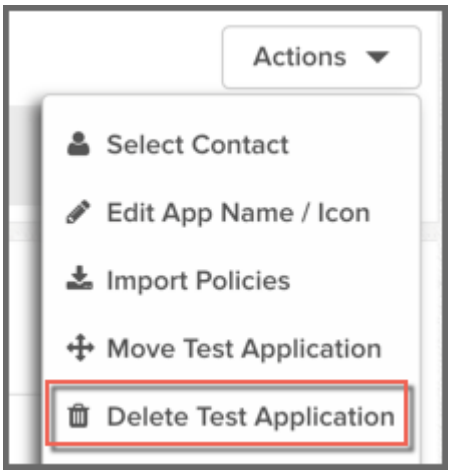
A Success message is displayed. If you get an error instead, the message lists the configuration discrepancies between the source and destination organizations that need to be resolved before the application can be moved.

⚠ The user who moves an application is assigned to the Owner role for the moved application. After a move, you may want to review user role assignments and notification settings, and adjust them as desired.

Deleting an Application

To delete an application:

1. Click the *Organization & Policies* icon on the IQ Server toolbar.Select an application in the sidebar.
2. In the sidebar, select the application you want to delete.
3. In the *Organization & Policies* area, click the *Actions* menu and choose *Delete <application name>*.



4. In the *Delete <application name>* dialog, click *Continue* to permanently delete the application or click *Cancel* to keep the application.

⊘ Deleting an application cannot be undone.

Viewing an Application

From anywhere on the IQ Server, you can view organizations and applications by clicking the Organization & Policy icon on the IQ Server toolbar. Once you are in the *Organization & Policies* area, you'll see all of your organizations displayed in the sidebar. Click the expander arrow next to an organization to view applications associated with that organization.

If you want to search for a specific organization or application, type a name into the filter box. After entering three characters, the sidebar is filtered automatically to display any matching applications and organizations. To remove the filter, click the "x" button next to the filter box.

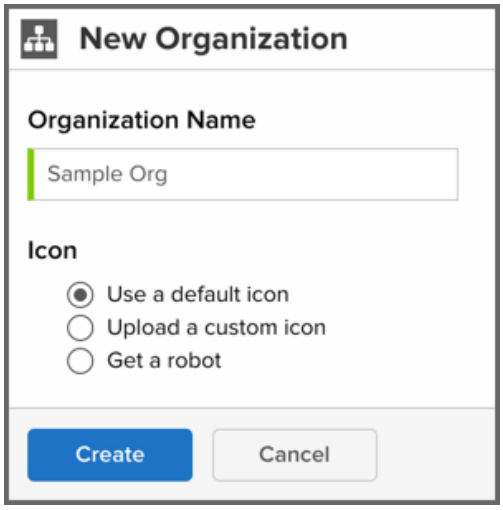
Organization Management

Creating Organizations

Organizations are created via the IQ Server. As a general rule with any activity on the IQ Server, make sure you have proper access. In the case of organization creation, only members of the Policy Administrator role have the permission to perform this actions.

To create an organization:

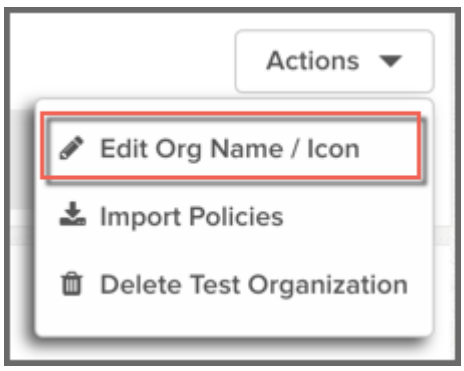
1. Click the *Organization & Policies* icon on the IQ Server toolbar.
2. Click the *Root Organization* icon in the sidebar.
3. Click the *New Organization* button.
4. In the *New Organization* dialog, set the following attributes:
    a. *Organization Name* - Enter a name into the text box.
    b. *Icon* - Select from three options:
        i. Default - Uses the default image.
        ii. Upload a custom icon - Click the *Upload Icon Image* button to navigate to an image file in PNG format and click Open. The image should be sized to 160 x 160 pixels. Images with different sizes will be scaled.
        iii. Get a robot - Click the *Get Another Robot* button to cycle randomly through a variety of robot images.
5. Click the *Create* button.



Editing an Organization

To edit an organization:

1. Click the *Organization & Policies* icon  on the IQ Server toolbar..
2. In the sidebar, select the organization you want to edit.
3. In the *Organization & Policies* area, go to the *Actions* menu and click *Edit Org Name / Icon*.
4. In the *Edit Organization* dialog, edit any the following attributes:
    a. *Organization Name* - Enter a different name into the text box.
    b. *Icon* - Select from three options:
        i. Use existing icon - To use the current image.
        ii. Upload a custom icon - Click the *Upload Icon Image* button to navigate to an image file in PNG format and click *Open*. The image should be sized to 160 x 160 pixels. Images with different sizes will be scaled.
        iii. Get a robot - Click the *Get Another Robot* button to cycle randomly through a variety of robot images.
5. Click the *Update* button to save changes.

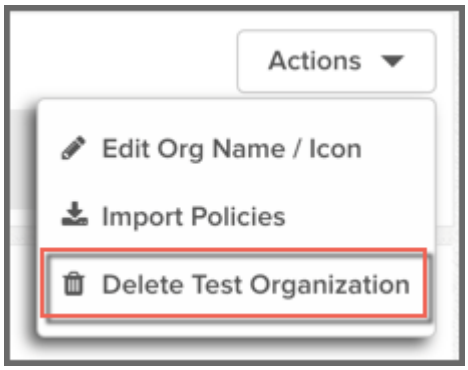


Deleting an Organization

To delete an organization:

1. Click the *Organization & Policies* icon  on the IQ Server toolbar.
2. In the sidebar, select the organization you want to delete.
3. In the *Delete <organization name>* dialog, click *Continue* to permanently delete the organization or click *Cancel* to keep the organization.

When you click **Continue** to delete an organization, this action cannot be undone.



Managing Automatic Applications

Overview

Nexus IQ Server 1.45 and above optionally support the automatic creation of applications when an application is analyzed for the first time. These applications are associated with a designated parent organization that can be configured using the configuration UI.

If automatic creation of applications is enabled when an application is analyzed by the Nexus IQ CLI[97], the application will be automatically created if it does not exist.

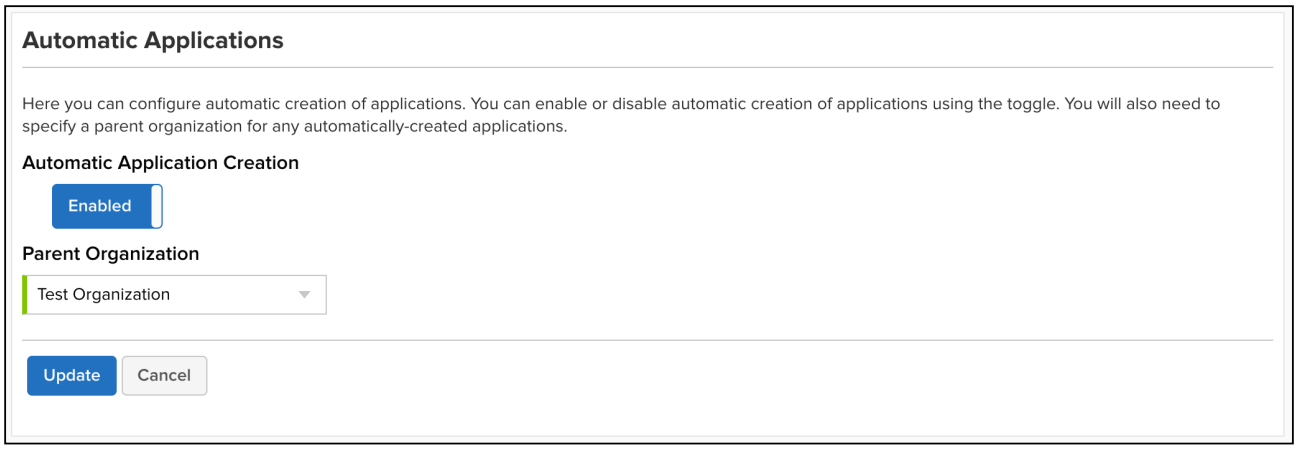
97 https://help.sonatype.com/display/NXI/Nexus+IQ+CLI

Configuring Automatic Application Creation

In order to enable this feature, it must be configured by a user with the Manage Automatic Application Creation permission. This new permission has been added to the built-in Policy Administrator role, but will need to be added manually to custom roles (see page 121) if desired.

To configure automatic applications, follow the instructions below:

1. Log into the IQ Server with a user that has the Manage Automatic Application Creation permission. This may be a user that has a Policy Administrator role.
2. Click the System Preferences icon  located in the top right of the header.
3. Choose *Automatic Applications* from the drop down menu. The Automatic Applications configuration page will now be displayed.
4. You can now make changes to the automatic applications configuration:
    a. The *Automatic Application Creation* toggle will allow you to enable or disable automatic application creation globally.
    b. The *Parent Organization* selects the organization that will be the parent of all subsequent automatically-created applications. You may need to create an organization (see page 142) if there are none available for selection.
5. Click the *Update* button to save your changes or *Cancel* to reset your local changes.



Changes to automatic application configuration will only impact those applications created after the changes have been saved. Previous automatically created applications will not be changed and will keep their original parent organization.

Policy Management

A policy is a set of rules that guide certain actions when conditions are met. It's what IQ Server uses to identify and prevent risk associated with open source, third-party, or proprietary components that may enter a repository or exist in an application.

Getting Started with Policies

To begin, there are several fundamental questions to ask yourself about risk and the components you use:

- What types of risks do you want to know about: security vulnerabilities, licensing problems, quality issues (like age or popularity), or something else?
- At what stage in the development lifecycle do you want to know about those risks?
- How severe do you think those risks are?
- What actions do you want to take? Receive a warning? Stop a build?
- Who should be notified of those risks? Particular individuals or whole groups?
- Do you want to constantly monitor inventoried components for new risk?
- How should the policies be applied in the system hierarchy? Globally, at the root organization level? More narrowly, at the organization level? Or even more narrowly, at the application level?

Reference Policy Set

Creating policies from scratch can be a complex and labor intensive process, and the Reference Policy Set will give you a head start.

When you start IQ server for the first time, the Reference Policy Set will be imported into the Root Organization automatically. You can also import the Reference Policy Set into an organization manually, as described in the section below.

The reference set contains policies for detecting and managing security, licensing, architectural, and popularity issues and includes some advanced policy features like application categories, component labels, and license threat groups. This policy set can help you gather information about the components used to build applications (including unknown and patched components), and understand how policy management will work for your environment.

Downloading the Reference Policy Set

You can download the reference policy set into an organization from here:

For IQ Server version **1.22 and newer**:

- reference-policies-v2.json[98]

For IQ Server version **1.21 or older**:

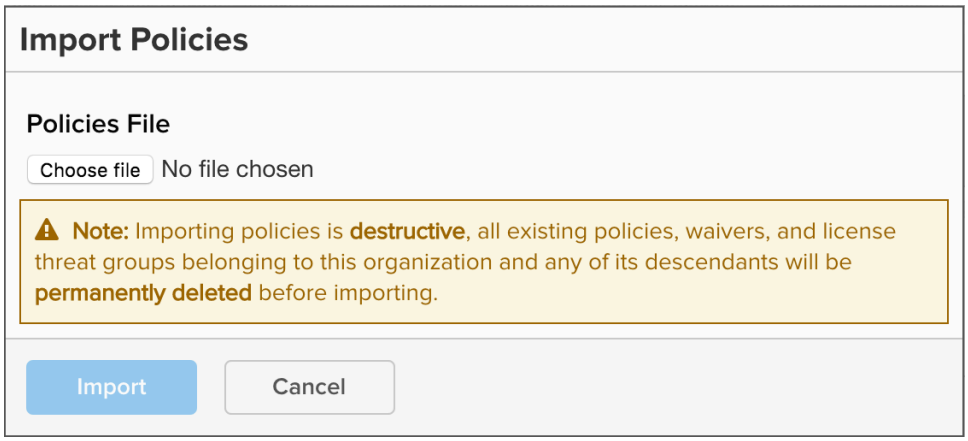
- reference-policies-v1.json[99]

Once the Reference Policy Set is downloaded, you can import it by following the instructions in the next section.

Importing Policies

After you acquire a policy file (in a .json format) such as the Reference Policy Set (see page 146), follow these steps to import it into IQ Server.

1. Log into IQ Server using an account that has permission to import policies into a specific organization (including the Root Organization). At a minimum, the account should be assigned to the Owner role of the organization.
2. Click the *Manage Applications and Organizations* icon on the IQ Server toolbar.
3. In the sidebar, click the organization into which you want to import the policy.
4. Click the *Actions* menu and select *Import Policies*.
   The Import Policy dialog is displayed as shown in the figure below.
5. Click the *Choose File* button and select the *policy .json* file in the file browser.
6. Click the *Import* button.



Rules for Importing Policies

If you want to import policies into an organization with existing policies (or application categories, component labels, and/or license threat groups), you should consider the following rules:

98 https://clm.sonatype.com/rest/referencePolicies/v2
99 https://clm.sonatype.com/rest/referencePolicies/v1

- Existing policies and waivers belonging to this organization and any of its descendants will be deleted during the import procedure.
- Importing policies also includes application categories, component labels, and license threat groups for which the following logic is used:
    - Application Categories - IQ Server attempts to match application categories against existing ones in a case-insensitive manner. This allows for updating the description or color of existing application categories, while preserving any current matching of categories between policies and applications.
    - Component labels - IQ Server attempts to match component labels against existing ones in a case-insensitive manner. This allows for updating the description or color of existing component labels, while preserving any triage effort already done to apply these labels to components. If your import contains component labels that aren't already present in the system, they will be created.
    - License Threat Groups - IQ Server will delete all existing license threat groups belonging to this organization and any of its descendants, and then import the new ones.

Related topics

- Understanding the Parts of a Policy (see page 148)
- Configuring Policies (see page 160)
- Continuous Monitoring of Apps (see page 163)
- Proprietary Component Configuration (see page 166)
- Component Labels (see page 168)
- License Threat Groups (see page 171)
- Application Categories (see page 178)
- Manual Application Evaluation (see page 182)
- Policy Violation Comparison Behavior (see page 183)

Understanding the Parts of a Policy

A policy is the set of rules or criteria used by IQ Server to evaluate components in your applications and repositories. It is made up of the following parts:

Summary

The summary section lets you define the Policy Name and Threat Level.

Policy Name

The Policy Name should be descriptive of the risk or violation you're trying to detect. In the text box, you can enter up to 60 characters: alphanumerics, underscores (_), periods (.), dashes (-), or spaces. The name you enter is used to identify the policy in IQ Server reports and views. To avoid confusion in your system hierarchy, it is recommended that you assign a unique name to every policy; try not to repeat names of policies created in other organizations.

Threat Level

The Threat Level is a subjective value placed on the perceived risk of a policy violation. Its main purpose is for sorting policy violations in IQ Server reports and views; the violations with the highest threat level appear first followed by those with lower threat levels. The Threat Level values are grouped by severity and identified by specific colors as shown in the table below:

Level	Color	Number
High	Red	8-10
Medium	Orange	4-7
Low	Yellow	2-3
Informational	Dark Blue	1
None	Light Blue	0

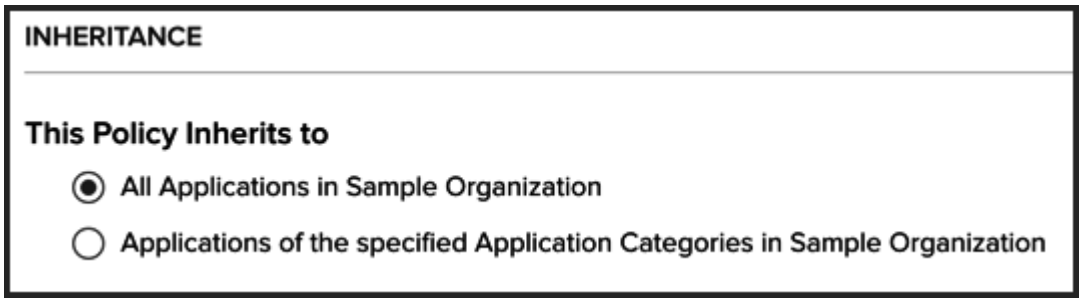
When setting the Threat Level, you should avoid causing unnecessary alarm for those who review policy violations. Select the lowest possible value that's helpful to you, such as an informational level (1) or low level (2-3). Save the high level values (8-10) for only the most critical violations, if used at all.

Inheritance

The Inheritance setting is available only for organizations (including the Root organization). It allows you to specify how a policy is implemented when there are multiple applications attached to a specific organization. There are two choices:

- All Applications in [organization] - The policy is applied to every application attached to the organization.
- Applications of the specified Application Categories in [organization] - The policy is applied only to applications that have specific application categories assigned to them. With this setting, you select which application categories to use.

The latter choice lets you tailor the implementation of a policy to applications with similar characteristics by using application categories. For more information on how to create and assign application categories, see Application Categories in the Advanced Policy Management chapter.



⚠ For policies at the Root Organization level, if you are a user of the Firewall solution, the All setting will include repositories as well applications. This will impact the policies used for the Audit and Quarantine features of IQ for Nexus Repository Manager.

Constraints

Constraints define the violations you want to detect. A constraint is essentially a container for conditions, and conditions are like the if part of an if/then statement. A policy must have at least one constraint, and each constraint must have at least one condition. When IQ Server evaluates your applications and the conditions of a constraint are met, then the policy is considered violated.

Constraints are made up of the following parts:

Item	Description
Constraint Name	This is a label for the constraint. It should describe the violation you want to detect, for example, High risk CVSS score or License needs legal review.
Conditions	IQ Server can detect many types of violations such as security vulnerabilities, licensing problems, quality concerns (like popularity or age), and more. To define a condition, you choose the type of condition you want from a built-in list and set any applicable parameters. See the section below for details on setting conditions.

Setting Conditions

Select a value for “Any or All" and then a condition and its parameters from the drop down menus. For more details on any of the conditions and their parameters, please see Application Composition Report (see page 193).

When setting conditions, you will also want to consider **Policy Type** (one of the available filters for the Dashboard (see page 0)). Based on the condition that is chosen, a policy type will be assigned. This is done automatically and can't be overridden. The following rules are used to determine a policy’s type:

- **Security** if it has any security conditions, it is considered a Security policy.
- **License** if it has any license conditions, it is considered a License policy.
- **Quality** if it has any age or popularity conditions, it is considered a Quality policy.
- **Other** if none of its conditions are of types mentioned above, it is considered to be of type Other.

A policy can only ever be of one type. In cases where a policy has conditions that meet more than one of the rules above, the order above dictates the type of policy. For example, if a policy has security and license conditions, it is considered to be of type Security.

Condition	Description
Any or All	Determines how constraints are evaluated. You can choose one of the following options: • Any - If any one of the conditions is met, then a policy violation is triggered. It is the equivalent of placing an or between each condition. This setting tends to produce a lot of policy violations. • All - If every condition is met, then a policy violation is triggered. This setting is the equivalent of placing an and between each condition. It tends to produce fewer policy violations.
Label	Verify if a specific component label is or is not assigned to a component.
License	Verify if the component license is or is not a specified license. If you've used the Component Information Panel to set a component's license status to Overridden, then any licenses designated as Declared or Observed are ignored. If a component's license status has not been overridden, then any occurrence (declared or observed) of the specified license is considered a match.
License Status	Verify if the status of a user-defined license is or is not one of the following values: Open, Acknowledged, Overridden, Selected, Confirmed.
License Threat Group	Verify if a component's license is or is not in a license threat group.
License Threat Group Level	Verify if the threat level of a component's license threat group is less than or equal or greater than or equal to a specified threat level value.
Security Vulnerability	Verify if a security vulnerability is present or absent in data sources searched by IQ Server.
Security Vulnerability Severity	Verify if a security vulnerability with a numeric severity is =, <, ⇐, >, or >= to a specified value.
Security Vulnerability Status	Verify if a component's security vulnerability status is or is not one of the following values: Open, Acknowledged, Not Applicable, Confirmed.
Relative Popularity (Percentage)	Verify if the relative popularity of a component's version (as compared to other versions of the same component) is =, <, ⇐, >, or >= to a specified percentage value.
Age	Verify if a component is older than or younger than a specified value.
Match State	Verify if the comparison of a component to known components is or is not a match in one of the following ways: Exact, Similar, or Unknown.

Coordinates	Verify if a component matches or does not match Maven or A-Name coordinates. For each type of coordinates, you enter specific attributes. You can use a wildcard (*) at the end of an attribute to broaden the search.
	
	Maven: You fill in a component's GAVEC, i.e. Group ID, Artifact ID, Version, Extension, and a Classifier. For example:
	
	`Group ID: org.sonatype.nexus`
	`Artifact ID: nexus-indexer`
	`Version: 1.0`
	`Extension: jar`
	`Classifier: sources`
	`Group ID: org.sonatype*`
	`Artifact ID: nexus-indexer`
	`Version: 1.*`
	`Extension: *`
	`Classifier:`
	
	A-Name: A-Name is short for Authoritative Name, an identifier created by Sonatype to identify components agnostic of the repository format. You fill in a Name, Qualifier, and Version, for example:
	
	`Name: log4net`
	`Qualifier: Framework 3.5`
	`Version: 2.0.5`
	`Name: log4net`
	`Qualifier:`
	`Version: 1.*`

Proprietary	Verify if a component is or is not considered proprietary.
Identification Source	Verify if the identification of a component is or is not one of the following: • Sonatype - When the identification is done based on IQ Server data sources. • Manual - When the identification is done based on a component claimed by you.

Actions

Policy actions allow you to designate an action to take when violations occur at a particular stage in the development lifecycle. For each stage, you can assign one of the following actions:

- *No Action* - This is the default setting.

- *Warn* - Policy violations are worthy of a warning.
- *Fail* - Policy violations are severe enough to potentially halt the development lifecycle.

If you connected IQ Server with an external tool, the action can have a direct effect on the tool. When an external tool requests a policy evaluation (of an application, repository or component), IQ Server provides policy violation information along with the action, which the tool may (or may not) implement. For example, if you set the Build stage to Fail in a policy, a CI tool (such as Bamboo, Jenkins, or Hudson) may stop the build of an application when that policy is violated. Similarly, in a different tool, if you set a stage to Warn, a warning message may be displayed or logged in a file when policy violations occur. For more details on using actions, see Usage Suggestions for Each Stage (see page 154).

To add actions to a policy:

1. In the *Organization & Policy* area, create a new policy or open an existing one for an organization or application.
2. In the *Policy* editor, click the **Actions** button to scroll to the Actions section.
3. Click the desired action — **No Action**, **Warn**, or **Fail** — at specific stage(s).
4. Click **Update** (or **Create**) to save the policy.

Usage Suggestions for Each Stage

Sonatype provides various tools and plugins to enable policy evaluation during each stage of development lifecycle:

- CLI
- Maven plugin
- Nexus Firewall
- IDE plugins
- CI plugins

Stage	Usage Suggestion
Proxy	Available only with the Nexus Firewall solution. Proxy refers to "Proxy Repository," or the point where components enter your repository manager. This is also referred to as the

	repository integration point. For more information on how to use the repository integration point, see the IQ for Nexus Repository Manager[100] topic. When setting actions, *Warn* will have no effect on what happens to components in the repository. However, if you have enabled Quarantine on a repository, and set the action to *Fail*, any new components added to the repository will be quarantined (unavailable via the Repository Manager). Quarantined components will not be available to your development team, including any attempt to build existing projects using those components.
Develop	The IDE plugin, CLI, and Maven plugin can use this stage. While actions and notifications can be configured for this stage, they may not affect the functionality of an IDE. CLI scans made with the Develop stage won't show up in the dashboard or the reporting view.
Build	This stage is typically used with CI plugins. As you manage policies, making necessary adjustments over time, it's best to take an approach that allows for your development teams to be eased into dealing with violations. For this reason, it's better to start by simply warning when the CI build for an application contains components that violate your policies.
Stage Release	Can be used with CI plugins or Nexus Repository Manager. Because this stage gives the opportunity to prevent an application from being released with components that have violated policies, setting the action for a Stage Release to *Fail* is recommended. This is especially true for any policies that may include risk associated with security and/or licensing.
Release	Can be used with CI plugins or Nexus Repository Manager. While there should be the closest scrutiny of policy violations at this point, it is recommended that you fail a release based on severe violations. Ideally, in most cases, you should be finding only new violations. If you have setup policy monitoring, it is a good idea to monitor your release stage, as this is likely the best representation of your production application.

100 https://help.sonatype.com/display/NXI/IQ+Server+and+Repository+Management

Operate	This stage is meant for integrations with deployment tools (e.g udeploy) or for use with the CLI in a deployment script. Scans made with the Operate stage won't show up in the reporting view.

Notifications

Notifications enable you to send a summary of new policy violations when they occur at a specific stage of development. Notifications are sent whenever an application is evaluated either manually (e.g. using the Evaluate Binary command in the Organization & Policy area) or automatically via any tool integrated into the IQ Server (e.g. Nexus IQ for Hudson/Jenkins 1.x) or if Continuous Monitoring is activated. The notifications can be delivered to email addresses or create a JIRA ticket. The emails are sent to individual addresses or users assigned to a particular role such as Owner or Application Evaluator. The JIRA tickets are created as a specified issue type for the selected JIRA project.

When a notification is sent, it will only display new violations found in the latest evaluation. If you find yourself not receiving notifications, verify there are new violations, as well as confirm you have configured your IQ Server SMTP settings. For information on SMTP settings in IQ Server, see the Email Configuration section of the IQ Server Setup chapter.

JIRA notifications are only available if a JIRA server is configured. See the JIRA Notifications section below.

When you have repository auditing setup, then notifications will be sent when a new component that violates policy enters your repository manager.

The initial repository audit and re-evaluations of policies on repositories do not send notifications.

To set notifications in a policy:

1. In the *Organization & Policy* area, create a new policy or open an existing one for an organization or application.
2. In the *Policy editor*, click the *Notifications* button to scroll to the Notifications section.
3. Provide recipient information:

a. Select a *Recipient Type*. If *Email*, then enter an email address. If *Role*, then select a user role from the list. For JIRA notifications, enter a project and select an applicable issue type.
b. Click *Add* to insert the recipient.

4. Click to select the stage(s) for which to send notifications to a recipient.

> ⚠ For the Continuous Monitoring stage, you must have monitoring activated for the application or a parent organization. To learn more about continuous monitoring, see Continuous Monitoring in Applications later in this chapter.

5. Click *Create* (or *Update*) to save the new policy.

> ✓ To remove a recipient, click the *Delete* button (looks like a trash can) for a particular recipient.

> ⚠ Be sure to select a stage for a recipient; if omitted, no notifications will be sent to the recipient.

Usage Suggestions for Notifications at Each Stage

Stage	Usage Suggestion
Proxy	Consider setting up notifications to inform repository owners or Nexus Repository Manager administrators that are responsible for safe-guarding components entering the organization. You can also view any policy violations that occur during this stage in the Repository Results.
Develop	Policy violations triggered by IDE-related activity generally do not send any notifications.
Build	Consider setting up notifications to inform owners, as well as developers.

Stage Release	If something fails, the development process can not move forward. Make sure to notify anyone who is responsible for the application's release and/or capable of researching and addressing any violations.
Release	Similar to Stage Release, make sure to notify anyone responsible for ensuring an application does not go into production with policy violations.
Operate	Typically the application owner, or anyone responsible for ongoing maintenance of an application in production should be notified.

JIRA Notifications

The JIRA notifications, as stated in the previous section, will create a JIRA issue when new policy violations are discovered during the development process. To create JIRA notifications, you must configure a valid JIRA server and credentials in your IQ Server's `config.yml`. Without this configuration you will not see the JIRA option in the Recipient Type drop-down. An example `config.yml` file is shown below:

```
# Notification JIRA settings.
jira:
  # The JIRA server address
  url: "https://127.0.0.1/"

  # The username used to connect to the JIRA server
  username: "anonymous"

  # The password used to connect to the JIRA server
  password: "guest"

  # Any custom fields that you wish to populate in the JIRA notification. Any
  # required fields must have default values configured here. Examples are shown below.
  #customFields:
  #   reporter:
  #     name: "username"
  #   labels:
  #     - test
  #     - bug
  #   environment: "dev"
  #   duedate: "2016-11-01"
```

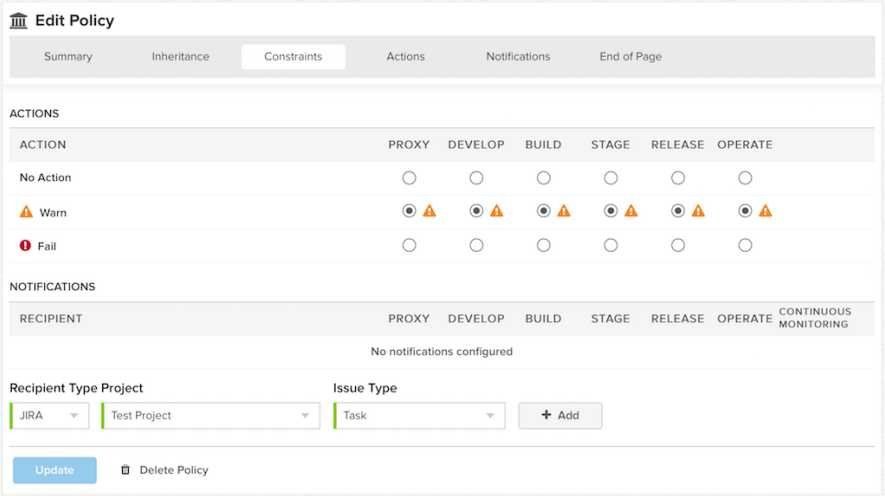
You may also configure any custom fields to be populated in the JIRA notification by utilizing the customFields section in the IQ Server `config.yml`. Any fields that are required on the issue type used for notifications must be configured this way otherwise the notification will fail.

A JIRA system user should be configured for integration with IQ Server. Any authenticated user of the IQ Server will be able to view the projects and applicable issue types available to the user configured in the config.yml. IQ Server users who can create and edit policy will be able to set up

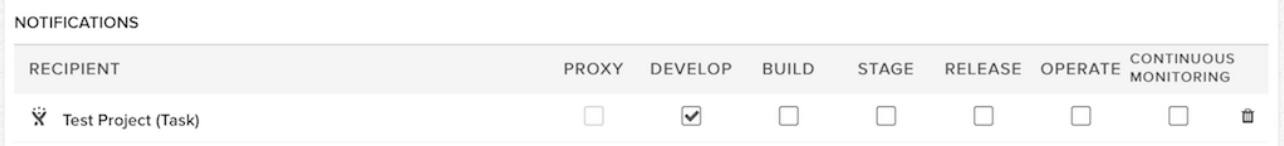
automated ticket creation for any project over which the configured JIRA user has authority to create tickets.

To configure JIRA notifications:

1. Select the Policy for which you will be notified when that policy is violated.
2. Select *JIRA* from the *Recipient Type* drop-down menu.
3. Select the *Project* and an *Issue Type*.
4. Click *Add* to add the notification.



Once you have created the notification, you can then choose at which stage(s) you would like to be notified.



In addition to having a valid JIRA server and credentials in the IQ Server's config.yml, if the JIRA issue type used for notifications has any required fields other than Summary and Description, those fields must have default values associated with them. Otherwise, IQ Server cannot create the issue in JIRA.

In JIRA 7, the Reporter field of an issue is required by default. In order to create JIRA notifications, you must either turn off the required setting, assign a default value for the issue type, or configure it using the custom fields section in the config.yml.

Configuring Policies

Viewing Policies

You can view policies, including those imported with the Reference Policy Set, (see page 146) by following these steps:

1. Log into IQ Server using an account that has permission to "View IQ Elements" for the specific organization or application. At a minimum, the account should be assigned the role of Owner or Developer for that organization or application.
2. Click the *Organization & Policies* button  on the IQ Server toolbar.
3. Select the desired organization or application in the sidebar.
4. Click the *Policies* button in the menubar near the top of the page to scroll to the *Policies* section.
5. Click the desired policy to view policy details.

Note that policies are grouped according to where they are located in the system hierarchy:

- *Local* - The policy was added at the level of the selected organization or application.
- *Inherited From [organization name]* - The policy was added at some level higher in the system hierarchy.

When you open an inherited policy, the view is read-only. You can expand collapsed sections in the view to see details, but you cannot make changes to the policy settings. For information on how to modify a policy, see Editing Policies (see page 162).

Creating Policies

Before you begin, you need to decide which level in the system hierarchy to use for new policies:

- Root Organization - Policies at this level are inherited by all organizations and applications. Use this level when you want to apply policies to every application and organization.
- Organization - Policies at this level are inherited by all applications attached to the organization. Use this level when you want to narrow the implementation of policies to a particular set of applications.
- Application - Policies at this level apply to an individual application only. Use this level when you want to apply policies to a single, unique application.

If you have access to the Audit and Quarantine features of IQ for Nexus Repository Manager, policies for your repositories are managed at the Root Organization level only. They do not require a specific application or organization.

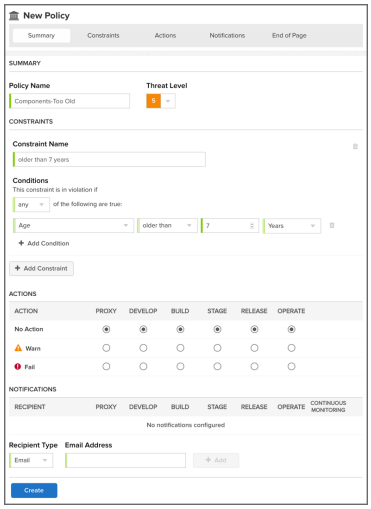
At the Root Organization and organization levels, you can use application categories to customize the implementation of policies across applications. Application categories provide a way to apply policies to a subset of select applications in an organization. For more details about application categories, see Application Categories in the Advanced Policy Management chapter.

Once you decide at which level to apply policies, you can proceed with creating custom policies. The overall process is only a few steps. However, the extent of customizable settings available to you can complicate the process. This section lists the basic steps for creating a policy, and includes links to more detailed information about each step in the Understanding the Parts of a Policy (see page 148) topic.

To create policies:

1. Log into IQ Server using an account that has permission to create policies in a particular organization or application (including the Root Organization). At a minimum, the account should be assigned to the Owner role of the organization or application.
2. Click the *Manage Applications and Organizations* icon  on the IQ Server toolbar.
3. In the *Policies* section, click *Add a Policy*.
   A New Policy view will be displayed.
4. Enter a name for the policy. For more details, see Policy Name (see page 148).
5. Select a threat level (from 10-0: 10 is the most severe threat, 0 is no threat). For more information, see Threat Level (see page 148).
6. If the policy is being created at the organization level, select which applications in the organization the policy should apply to: all applications or only applications with selected application categories. If the latter, then click the specific application categories to select them. For more details, see Inheritance (see page 132). Note that this setting is not available when creating a policy for an application.
7. Create a constraint with conditions. For detailed information, see Constraints and Conditions (see page 148).
8. Add actions and/or notifications at a desired stage in the development lifecycle. For more information, see Actions and Notifications (see page 148).
9. Click **Create** to save the policy.

After at least one policy is created (or imported), you can run an evaluation of an application to gather intelligence about its components and identify any vulnerabilities. The evaluation results, which include policy violations, are displayed in the Application Composition Report. For more information, see the Manual Application Evaluation (see page 182) and the Application Composition Report (see page 193) sections.

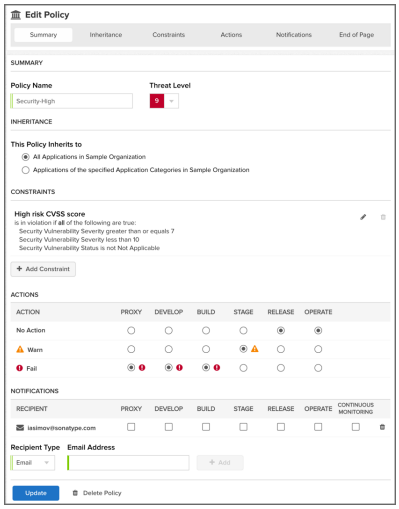


Editing Policies

At some point, you may want to edit an existing policy. For example, you'd like to modify a policy in the Reference Policy Set (see page 146) to suit the needs of your development team. The process for editing a policy is almost the same as creating one; it's only a few steps. However, the extent of customization you can do may make the process more complicated. This section lists the overall steps for editing a policy, and includes links to more detailed information in the Understanding the Parts of a Policy (see page 148) topic.

To edit policies:

1. Log into IQ Server using an account that has permission to edit policies in a particular application or organization. At a minimum, the account should be assigned to the *Owner* role of the organization or application.
2. Click the *Manage Applications and Organizations* icon  on the IQ Server toolbar.
3. In the sidebar, select the organization or application in which the policy was created.
4. In the *Policies* section under *Local*, click the policy you want to edit. If the policy is listed in an Inherited From section, then it was created at a higher level in the system hierarchy; you must go to the level in which it was created to edit it.
5. In the *Edit Policy* view, you can change the following settings:
    a. Enter a new name.
    b. Select a different threat level.
    c. If at the organization level, change which applications the policy applies to: all applications or only applications with selected application categories. If the latter, then click the specific application categories to select them.
    d. Add or modify a constraint with conditions.
    e. Add or modify actions and/or notifications.
6. Click *Update* to save the policy changes.



Deleting Policies

To delete policies:

1. Log into IQ Server using an account that has permission to delete policies in a particular application or organization. At a minimum, the account should be assigned to the *Owner* role of the organization or application.
2. Click the *Manage Applications and Organizations* icon on the IQ Server toolbar.
3. In the sidebar, select the organization or application in which the policy was created.
4. In the *Policies* section under *Local*, click the policy you want to delete. If the policy is listed in an *Inherited From* section, then it was created at a higher level in the system hierarchy; you must go to the level in which it was created to delete it.
5. In the *Edit Policy* view, click the *Delete Policy* button.
6. In the confirmation dialog box, click *Continue* to permanently delete the policy or *Cancel* to keep the policy.

Once you delete a policy, the action cannot be undone.

Continuous Monitoring of Apps

At times, you may want to be notified when applications no longer in development (or being built on a regular basis) have components that violate a policy. For example, you'd like to learn of any security vulnerabilities or licensing issues that may arise after applications are deployed. Continuous Monitoring lets you use existing policies with notifications to constantly watch (once a day) for new violations at a specific development stage (such as Release).

Use Continuous Monitoring judiciously. If too many messages are sent for minor violations, it could result in notification fatigue for your development team. You may want to limit the monitoring to policies that detect high risk violations, like security vulnerabilities or license concerns.

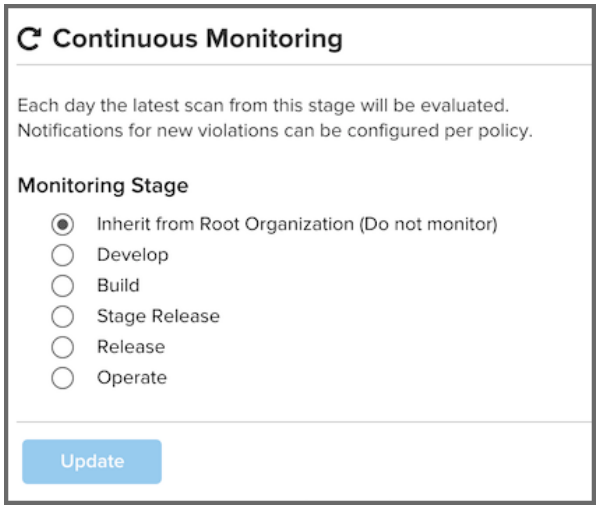
Setting up Continuous Monitoring is a two-step process. First, you turn on Continuous Monitoring at the organization or application level, and specify which stage of the development lifecycle to monitor. Second, you turn on Continuous Monitoring at the policy level by creating a notification and selecting Continuous Monitoring in a policy. Each of these steps is described in more detail below.

Step 1: The Application or Organization Level

Continuous Monitoring, by default, is turned off for the Root Organization. Because all organizations and applications inherit policy settings from the Root Organization, it is turned off for those entities as well. You can turn on Continuous Monitoring for individual applications, or an organization (the parent) and all of its associated applications (the children). You also specify which stage of the development lifecycle to monitor.

To turn on Continuous Monitoring for an application or organization:

1. Click the *Organization & Policies* icon  on the IQ Server toolbar.
2. In the sidebar, select the organization or application whose policies you want to monitor.
3. In the *Policies* section, under *Continuous Monitoring*, click the chevron next to either *Do Not Monitor* or *Inherit from [parent] (Do not monitor)*.
4. In the *Continuous Monitoring* view, click the desired stage.
5. Click *Update* to turn on Continuous Monitoring.



Step 2: The Policy Level

When you turn on Continuous Monitoring at the policy level, you are identifying who should receive an email message when a violation of the current policy occurs at a particular development stage (specified in Step 1) whenever an evaluation is performed.

To turn on Continuous Monitoring in a policy:

1. In the *Organization & Policy* area, create a new policy or open an existing one for an organization or application.
2. In the *Policy* editor, click the *Notifications* button to scroll to the *Notifications* section.
3. Make sure the Notifications Recipient list contains the desired email address to use for policy violation notifications. If necessary, add a new recipient.
4. For the desired email address, click *Continuous Monitoring* to select it.
5. Click *Update* to save the policy.

If you perform Step 1, but omit Step 2, no notifications of policy violations will be sent when a Continuous Monitoring evaluation is run. You must perform Step 1 and Step 2 for Continuous Monitoring to work properly.

Turning off Continuous Monitoring

To turn off Continuous Monitoring:

1. Click the *Organization & Policies* icon on the IQ Server toolbar.
2. In the sidebar, select the desired organization or application.
3. In the *Policies* section, under *Continuous Monitoring*, click the chevron next to the stage that's being monitored.
4. In the *Continuous Monitoring* view, click whichever of the following options is displayed:
    a. For the Root Organization, click *Do not monitor*.
    b. For other organizations and applications, click *Inherit from [parent] (Do not monitor)*.
5. Click *Update* to save your change.

If an organization or application's parent has monitoring enabled, there is no way to disable its monitoring and the option will read Inherit from [parent] (Monitored Stage). Monitoring must be disabled throughout an organization or application's hierarchy in order to disable it.

Setting the Notification Time

Once Continuous Monitoring is turned on, you may want to consider the time of day that notifications are sent. By default, they are sent at 0000 hours or 12:00 a.m. (per IQ Server time). You can change the notification time setting in IQ Server's config.yml file as follows:

```
# Hour of the day(0-23) to schedule Policy Monitoring execution. The
default is midnight.
policyMonitoringHour: 0
```

Proprietary Component Configuration

Proprietary components are unique or internal to your organization. When you evaluate an application that uses proprietary components, IQ Server is unlikely to find data about those components; they are probably listed under "Unknown" on the "Policy Violations" tab in the Application Composition Report. However, you can configure IQ Server to recognize those components as proprietary.

When you configure proprietary components, you use system hierarchy levels to set the scope for identifying the components:

- Root Organization - Identifies proprietary components in every organization and application.
- Organization - Identifies proprietary components in a particular set of applications.
- Application - Identifies proprietary components in a single application.

You also specify a string search pattern called a proprietary component matcher that IQ Server uses to find proprietary components. If matching components are found, they are displayed under *Proprietary* on the *Policy Violations* tab in the Application Composition Report. There are two types of proprietary component matchers: *Package* and *Regular Expression*, which are described below.

Package Matchers

For *Package*, you specify a package name, for example, com.sonatype. In this case, all components that contain a package com/sonatype will be marked as proprietary. You should be as specific as possible, for the provided package is compared greedily against your scanned binaries. For instance, com.sonatype will match all of the following content locations:

- com/sonatype
- com/sonatype/anything
- com/sonatype/anything/more
- shaded/and/relocated/com/sonatype
- shaded/and/relocated/com/sonatype/anything

On the other hand, the following locations will not be matched:

- org/sonatype
- com/sonatypestuff
- com/sonatypestuff/anything

Regular Expression Matchers

For *Regular Expression*, you specify a regular expression that will be compared against the paths of all files scanned. If a file is found in the path, it is flagged as proprietary. For example, **`test\.zip`** will recognize anything in the top level directory named test.zip as proprietary. If you wanted to find test.zip nested anywhere in the scanned binaries, use **`.*/test\.zip`**.

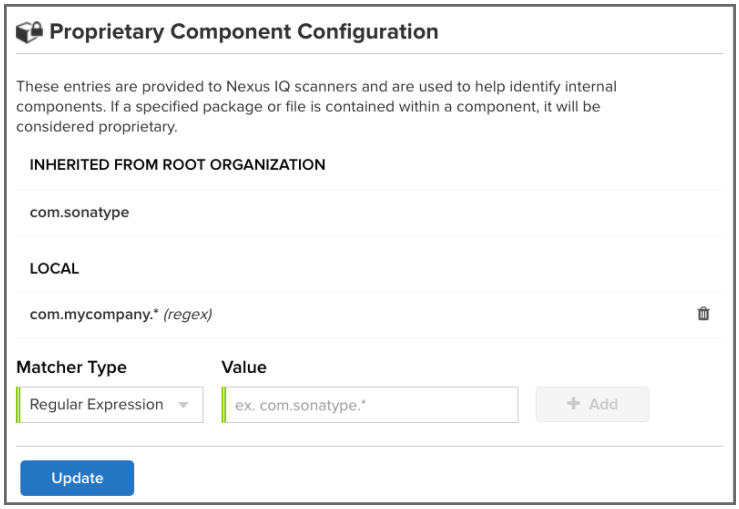
> Occurrences inside an identified archive will make the binary proprietary as well. For example, if a proprietary .zip is found inside a .jar, the .jar is also considered proprietary.

For more information on regular expressions, see Oracle's Java documentation[101].

To configure proprietary components:

1. Click the *Organizations & Policies* icon  on the IQ Server toolbar.
2. In the sidebar, select the desired organization or application.
3. In the *Policies* section, under *Proprietary Component Configuration*, click the chevron next to the number of matchers (local and/or inherited).
4. In the *Proprietary Component Configuration* view, add or remove matchers as desired. To add, select a Package or Regular Expression matcher type and enter a string search value. To remove, click the *Delete* icon (looks like a trash can) for items in the Local section.
5. Click *Update* to modify IQ Server's list of proprietary component matchers for the selected organization or application.

101 http://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html



Usage Suggestions for Proprietary Components

Once proprietary components are configured, you can use a policy to prevent them from triggering policy violations. There is an example of this in the Reference Policy Set (see page 146); the *Component-Unknown* policy has the following constraint:

The policy is in violation if *all* of the following are true:

- Match State is Unknown
- Proprietary is false

This constraint excludes proprietary components from triggering policy violations.

Component Labels

Component labels are actually one of the more powerful features of policy management, and they should have a familiar look, since you've likely used other systems that employ a sort of tagging or labeling.

Essentially, component labels are metadata. More specifically a component label is metadata that is assigned to a component within the context of a particular application or organization. Labels can assist with identifying components you want to review, approve, or even avoid altogether. We call this component label assignment.

When component labels are assigned, this is an action that takes place in the application composition report. Before it can be assigned though, a label needs to exist for a particular organization or application.

As we learned in our Organization and Application Management chapter, inheritance (see page 132) plays a big role in policy. The same thing is true for labels, in that if a component label is created in an organization, any application attached to that organization will also have the label available for use when assigned. In fact, the

system will prompt you to choose the scope (organization or application) a label should exist in when it is assigned.

You can customize a policy to use a component label as a condition when IQ Server evaluates applications. For more information about policies and creating conditions, see Understanding the Parts of a Policy (see page 148).

Viewing a Component Label

To view a component label:

1. Click the *Organization & Policies* icon  on the IQ Server toolbar.
2. Select an organization or application in the sidebar. A page of customizable settings is displayed.
3. Click *Component Labels* in the menu bar at the top of the page to scroll to the *Component Labels* section.

The Component Labels section displays two types of labels:

- *Local* - Component labels with a scope that's specific to the selected organization or application.
- *Inherited* - Component labels derived from an organization that's higher in the system hierarchy than the currently selected organization or application.

Creating a Component Label

To create a component label:

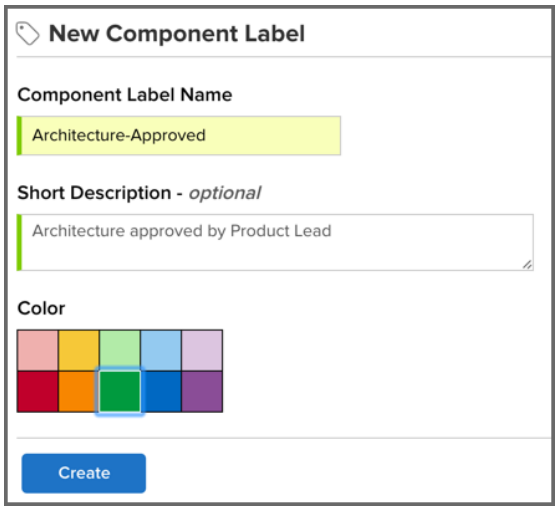
1. Click the *Organization & Policies* icon  on the IQ Server toolbar.
2. In the sidebar, select the organization or application in which the component label will be used.
3. In the *Organization & Policies* area, click *Component Labels*.
4. In the *Component Labels* section, click the *Add a Label* button. The *New Component Label* editor is displayed.
5. In the *New Component Label* editor, set the following attributes:

a. *Component Label Name* - Enter a name for the component label that is easily identified.
b. *Short Description* - Enter a description that provides additional information about the component label.
c. *Color* - Select a desired color for the component label.

6. Click the *Create* button to add the component label to the selected organization.

A few things to remember:

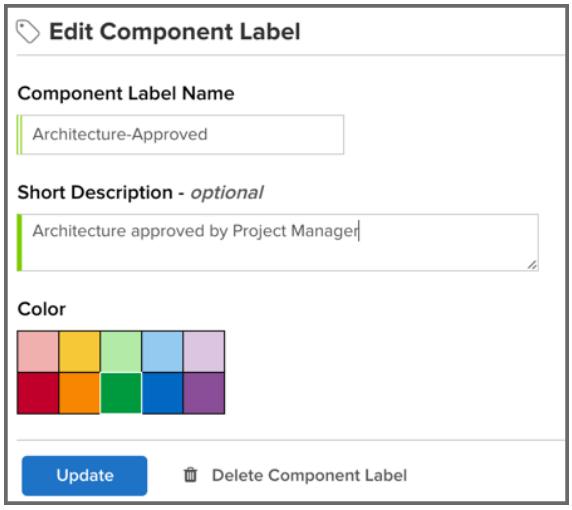
- An organization's component labels can be seen by any of its applications, the reverse is not true.
- Component labels can only be edited (or deleted) at the level they were created.



Editing a Component Label

To edit a component label:

1. Click the *Organization & Policies* icon  on the IQ Server toolbar.
2. In the sidebar, select the organization or application in which the component label was created. The component label is displayed in the *Component Labels* section under the *Local* heading, and has a chevron in its row to indicate it's editable.
3. Click the component label you want to edit. The *Edit Component Label* dialog is displayed.
4. In the *Edit Component Label* dialog, you can change the following attributes:
    a. *Component Label Name* - Enter a different name for the component label.
    b. *Short Description* - Enter a description that provides additional information about the component label.
    c. *Color* - Select a desired color for the component label.
5. Click the *Update* button to save the component label to the selected organization or application.



Deleting a Component Label

To delete a component label:

1. Click the *Organization & Policies* icon on the IQ Server toolbar.
2. In the sidebar, select the organization or application in which the component label was created. The component label is displayed in the *Component Labels* section under the *Local* heading, and has a chevron in its row to indicate it's editable.
3. Click the component label you want to delete. The *Edit Component Label* dialog is displayed.
4. In the *Edit Component Label* dialog, click the *Delete Component Label* button. A *Delete Label* alert dialog is displayed.
5. In the *Delete Label* dialog, click *Continue* to delete the component label or *Cancel* to keep the component label.

When you delete a component label, the action cannot be undone.

License Threat Groups

License threat groups, are simply groups of licenses, broken into categories of severity for the various types of licenses. They can help you to achieve your goals related to enforcing the usage of components with licensing that matches the scope of your application.

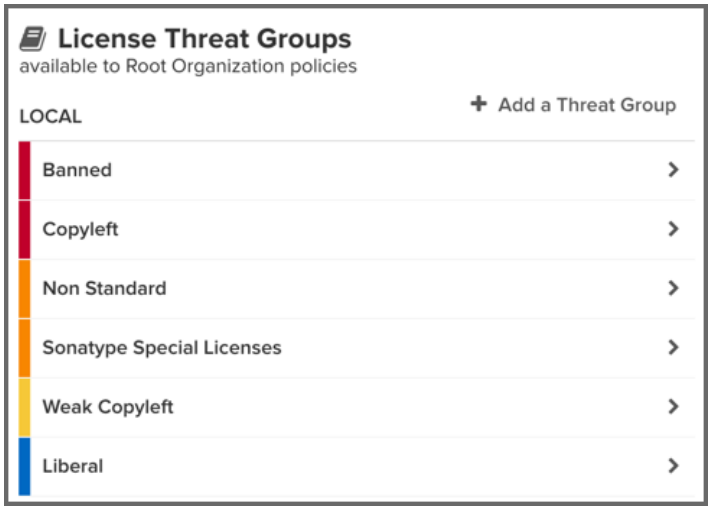
Their primary purpose is to serve as the data points for the License section (see page 209) of the Application Composition Report. Moreover, they are a way to group risk, associated with licensing.

You can customize a policy to use a license threat group (or an unassigned license threat group) as a condition when IQ Server evaluates applications. For more information about policies and creating conditions, see Getting Started with Policies (see page 146).

Viewing a License Threat Group

To view a License Threat Group:

1. Click the *Organization & Policies* icon on the IQ Server toolbar.
2. Select an organization in the sidebar. A page of customizable settings is displayed.
3. Click *License Threat Groups* in the menu bar at the top of the page to scroll to the *License Threat Groups* section. The list of License Threat Groups is organized by where the groups are defined: *Local* for the currently selected organization or *Inherited From* for an organization higher in the system hierarchy.



The following license threat groups are included by default for the root organization:

Banned	Any licenses that should not be permitted in any circumstances. This license threat group contains the AGPL licenses by default.
Copyleft	Strong copyleft licenses go a step further from weak copyleft licenses and mandate that any distributed software that links or otherwise incorporates such code be licensed under compatible licenses, which are a subset of the available open-source licenses. As a result, these licenses have been called viral.
Liberal	These licenses allow you to do almost anything conceivable with the program and its source code, including distributing then, selling them, using the resultant software for any purpose, incorporating into other software, or even converting copies to different licenses, including that of non-free (so-called “proprietary”) software.

Non Standard	Something out of the ordinary (e.g. If we ever meet, give me a beer license).
Sonatype Special Licenses	A license threat group for identifying situations where Sonatype has been unable to determine the license of a component.
Weak Copyleft	Free software licenses that mandate that source code that descended from software licensed under them, will remain under the same, weak copyleft, license. However, one can link to weak copyleft code from code under a different license (including non-open-source code), or otherwise incorporate it in a larger software. Otherwise, weak copyleft licenses allow free distribution, use , selling copies of the code or the binaries (as long as the binaries are accompanied by the (unobfuscated) source code), etc.

Consult with your legal department for EXACT definitions. Information provided above is from the following reference[102].

Creating a License Threat Group

An important aspect of license threat groups is that each one also has a threat level, just like policy (from zero signifying no threat all the way up to 10). Unless you have specific legal recommendation / council, the default license threat groups will suffice, especially in the beginning.

If you desire, you can edit these default groups, or create entirely new ones. When creating license threat groups, keep in mind that they will be inherited from the organization to all associated applications.

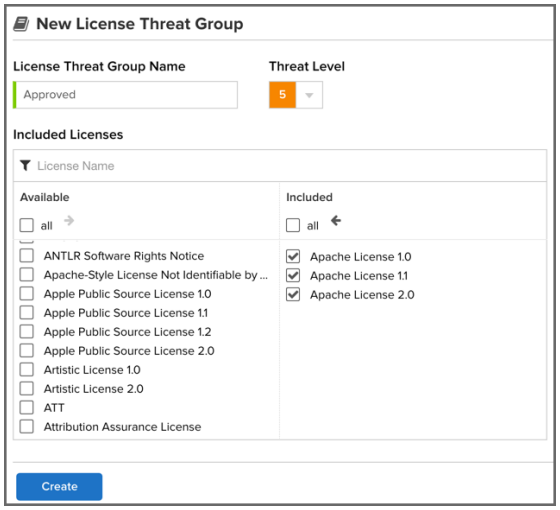
To create a license threat group:

1. Click the *Organization & Policies* icon  on the IQ Server toolbar.
2. In the sidebar, select an organization.
3. Click *License Threat Groups* in the menu bar at the top of the page to scroll to the *License Threat Groups* section.
4. Click the *Add a Threat Group* button.
5. In the *New License Threat Group* editor, set the following attributes:
    a. *License Threat Group Name* - Enter a name for the license threat group that is easily identifiable.

102 http://www.shlomifish.org/philosophy/computers/open-source/foss-licences-wars/foss-licences-wars/types-of-licences.html#weak-copyleft-licences

b. *Threat Level* - Select a number for the threat level that this group of licenses represents.
c. *Included Licenses* - Type a string of characters in the filter box or scroll the *Available* list to locate desired licenses by name.
    i. In the *Available* column on the left, select a license in the list, then click the right arrow button to move the license to the Included column on the right.
    ii. If you accidentally add a wrong license, select the license in the *Included* column, then click the left arrow to return it to the *Available* column.

6. Click *Create*.



As of IQ Server 1.20, license threat groups are no longer created at the application level. If you previously had license threat groups in your applications, you can still edit them, but we encourage you to migrate those license threat groups up to the organization.

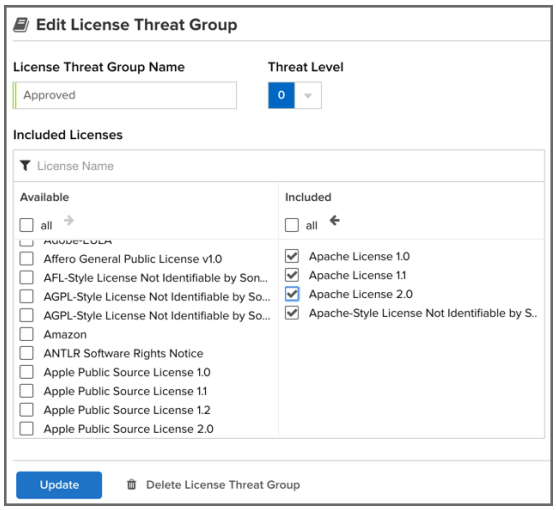
Editing a License Threat Group

To edit a license threat group:

1. Click the *Organization & Policies* icon  on the IQ Server toolbar.
2. In the sidebar, select the organization (or application, if created prior to IQ Server 1.20) in which a license threat group was created.
3. Click *License Threat Groups* in the menu bar at the top of the page to scroll to the *License Threat Groups* section.
4. In the list of License Threat Groups, click the one you want to edit (it has a chevron in its row to indicate it's editable).
5. In the *License Threat Group* editor, you can set the following attributes:

a. *License Threat Group Name* - Enter a different name for the license threat group that is easily identifiable.
b. *Threat Level* - Select a number for the threat level that this group of licenses represents.
c. *Included Licenses* - Type a string of characters in the filter box or scroll the columns of licenses to locate desired licenses by name.
    i. To add a license, select the license in the *Available* column on the left, then click the right arrow button to move the license to the *Included* column on the right.
    ii. To remove a license, select the license in the Included column, then click the left arrow to return it to the *Available* column.

6. Click *Update*.



Deleting a License Threat Group

To delete a license threat group:

1. Click the *Organization & Policies* icon on the IQ Server toolbar.
2. In the sidebar, select the organization in which a license threat group was created.
3. Click *License Threat Groups* in the menu bar at the top of the page to scroll to the *License Threat Groups* section.
4. In the list of License Threat Groups, click the one you want to delete (it has a chevron in its row to indicate it's editable).
5. In the *License Threat Group* editor, click the *Delete License Threat Group* button. A warning message is displayed.
6. Click *Continue* to permanently remove the License Threat Group or *Cancel* to keep it.

Understanding License Type

License Threat Groups (see page 171) allow us to categorize licenses, based on the potential threat they provide. In addition, there are other "license types" which we'll further describe here, as they can enhance your understanding of the threat, or alter the results in your policy violations. Please see Component License Information (see page 209) for basic license type descriptions.

Types of License Identification

There are three types of license identification:

- *Declared License* relates to the license information supplied as part of the project which develops the component, typically declared at the root of the component, for example Java as part of the pom.xml.
- *Observed License* relates to the license information observed as part of Sonatype's research. This may be as a result of a license embedded within the source code, or documented elsewhere within the project metadata, or website etc. This may or may not be the same as the Declared license. This may differ as a result of the re-use of code from another project, a developer deciding to license their code independently of the project, or other reasons.
- *Effective License* relates to the combination of the Declared and Observed License types, which may lead to multiple options.

License Policy Options

When creating a license policy, there are four conditions that are available to process license risk:

> License
>
> License Status
>
> License Threat Group
>
> License Threat Group Level

License relates to the license type, where you are referencing the effective license(s).

Sonatype continually updates the license information within the product. For expediency throughout the user interface, there is a combination of short name and long name used in IQ server. For example, the Component Information Panel (CIP) and policy definition tend to the short name, while the license threat group condition creations use the long name. A mapping of common licenses can be found at https://spdx.org/licenses/.

In addition to actual license types, additional options include:

License	is	Non-Standard
License	is	Not Declared
License	is	Not Provided
License	is	Not Supported
License	is	No Sources
License	is	No Source License

Long Name	Short Name	Description
Not known at this time or there isnt one	Non-Standard	We have a non standard threat group.
No licenses declared in component descriptor	Not Declared	You have declared no license, this different from the **Not Provided** license type, assuming that we may have embedded some license text but included no specific license type (i.e. Apache or GNU etc). This is valid across all ecosystems on a per component basis.
No License Information was Provided	Not Provided	Will appear when the license is actually null, this is unique to claimed components; in addition this may also apply as new components are being processed by Sonatype.
Not Supported	Not Supported	Only seen as an Observed License attribute for non-java eco system where we dont ingest source license details.
No licenses found in sources	No Source License	This means that you have a source, but there are no license statements (i.e. no Observed License) otherwise we see an observed license.
No sources provided with component	No Sources	This means the component has no source code. This is valid across all eco-systems.

Application Categories

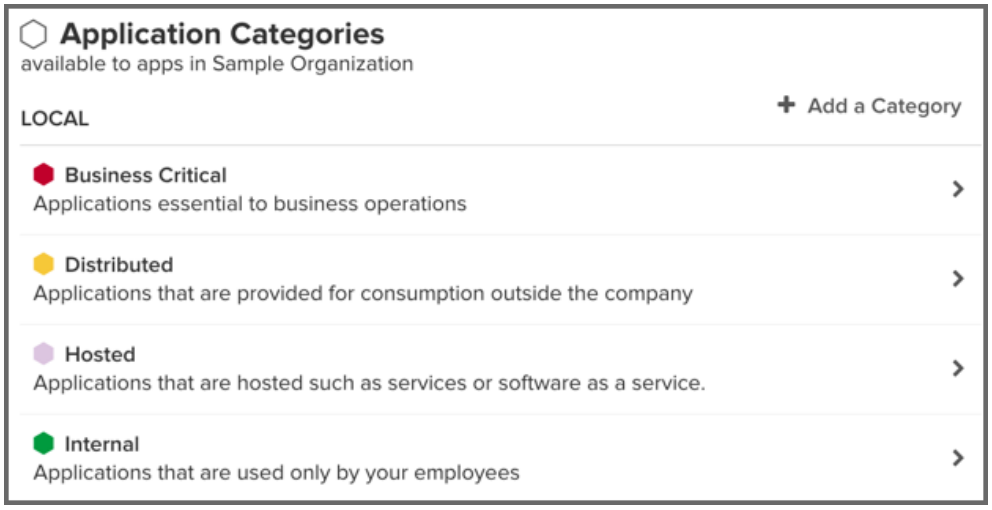
In any given business, you could have hundreds, maybe even thousands of applications. Even if you are just getting started, it's likely you have a handful of applications. However, as unique as applications can be, they tend to share some similarities.

For example, you might have applications that process or store sensitive information, maybe even personally identifiable information for your users. Since attacks are often aimed at these types of applications, you will definitely want to make sure your policies that identify high and critical threat security vulnerabilities are included during the evaluation of these types of applications.

Unfortunately, especially as the number of applications in your business increases, identifying an application by name may not be helpful. To address this, *application categories* provide a way to quickly identify characteristics of an application.

Using specific text and color, an application category can help group particular applications with similar attributes. While an application category can ultimately be anything you want, and attached to any application, you will want to take a much more thought-out approach, similar to what is recommended for labels.

As we will see later, in order to maximize the benefits application categories can offer, you will want to take advantage of category matching between policies and applications. For now though, let's see how to create, edit, delete, and apply application categories.

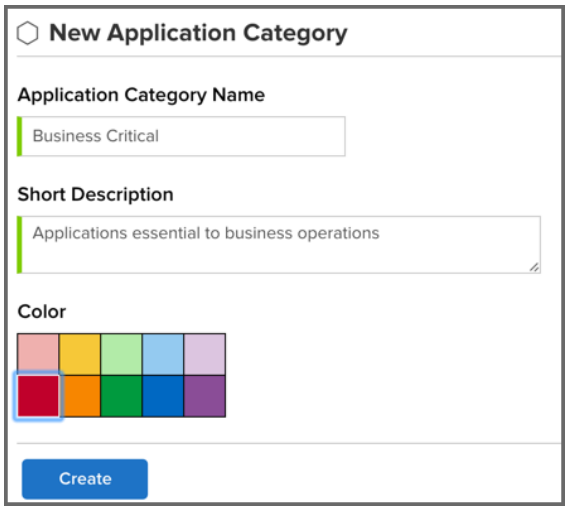


Creating Application Categories

Application categories are created, edited, and deleted at the organization level and then assigned individually to each application.

To create an application category:

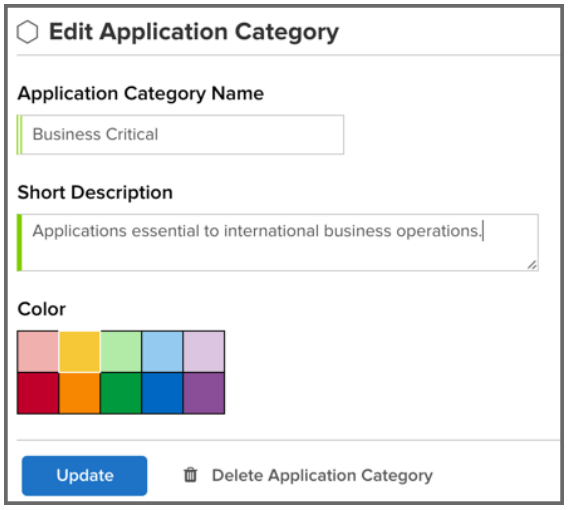
1. Click the *Organization & Policies* icon  on the IQ Server toolbar.
2. In the sidebar, select an organization.
3. In the *Application Categories* section, click *Add a Category*. The *New Application Category* dialog is displayed.
4. In the *New Application Category* dialog, set the following attributes:
    a. *Application Category Name* - Enter a name that is easily identified for it will be used to match an application to corresponding policies.
    b. *Short Description* - Enter a description that provides additional information about the category.
    c. *Color* - Select a desired color for the category.
5. Click the *Create* button to add the application category to the selected organization.



Editing an Application Category

To edit an application category:

1. Click the *Organization & Policies* icon  on the IQ Server toolbar.
2. In the sidebar, select the organization in which the application category was created. The application category is displayed in the *Application Categories* section under the *Local* heading, and has a chevron in its row to indicate it's editable.
3. Click the application category you want to edit. The *Edit Application Category* dialog is displayed.
4. In the *Edit Application Category* dialog, you can change the following attributes:
    a. *Application Category Name* - Enter a different name.
    b. *Short Description* - Enter a description that provides additional information about the category.
    c. *Color* - Select a desired color for the category.
5. Click the *Update* button to save your changes to the application category.



Deleting an Application Category

To delete an application category:

1. Click the *Organization & Policies* icon on the IQ Server toolbar.
2. In the sidebar, select the organization in which the application category was created. The application category is displayed in the *Application Categories* section under the *Local* heading, and has a chevron in its row to indicate it's editable.
3. Click the application category you want to delete. The *Edit Application Category* dialog is displayed.
4. In the *Edit Application Category* dialog, click the *Delete Application Category* button. A *Delete Category* alert dialog is displayed. If there are applications assigned to the application category, they will be listed.
5. In the *Delete Category* dialog, click *Continue* to delete the application category or *Cancel* to keep the application category.

When you delete an application category, the action cannot be undone.

You cannot delete an application category that's used in a policy to affect policy inheritance. You must first remove the application category from the policy, and then delete the application category.

Assigning an Application Category

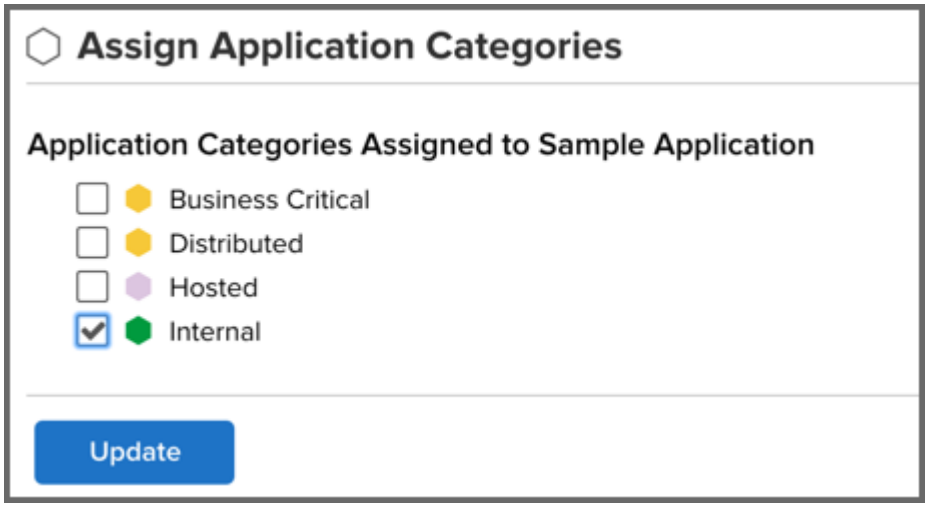
In most cases, the people assigning application categories may be different from those creating them. It is important though to understand that while application categories are provided to identify characteristics of an application, a more important usage is to provide a way for policy managers to create specific policies

that consider those application characteristics. For this reason, when assigning an application category, your application may be evaluated by a specific set of policies. This is a good thing, but it also makes the use of application categories an act that requires careful consideration.

To assign an application category to an application:

1. Log in to the IQ Server using a user account that's assigned to a role with Owner-level permissions for the application. The built-in Owner role has owner-level permissions by default.
2. Click the *Organization & Policies* icon on the IQ Server toolbar.
3. In the sidebar, select an application.
4. In the *Application Categories* section, click *Assign App Categories*. The *Assign Application Categories* page is displayed.
5. In the list of assigned application categories, select the application categories you want to assign to the selected application.
6. Click *Update* to save your selection(s).

There must be at least one application category defined before you can assign any application categories. For more information, see Application Categories (see page 178).



Once application categories are created and assigned, you can use them to apply policies to a subset of applications in an organization through inheritance. For more information about policy inheritance and application categories, see the Basic Policy Management chapter.

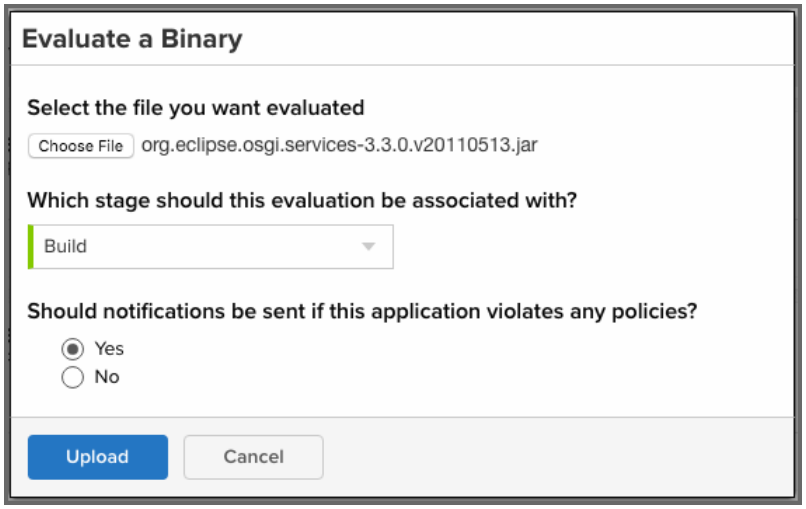
Manual Application Evaluation

When you evaluate an application, IQ Server generates a report that gives you a baseline of your application's health. Before you can initiate an evaluation, you need the following items:

- At least one organization and one application. For information on creating these entities, see Organization and Application Management (see page 131).
- View IQ Elements and Evaluate Applications permissions for the application you wish to evaluate. For details about setting permissions, see Role Management (see page 121).

To evaluate an application:

1. Log in to IQ Server as a user assigned to the Policy Administrator or Owner role for the application.
2. Click the *Organization & Policies* icon  on the IQ Server toolbar.
3. Select an application in the sidebar.
4. Go to the *Actions* menu, and click *Evaluate Binary*.
5. In the Evaluate a Binary dialog, do the following:
    a. Click the *Choose File* button and then navigate to the file you want evaluated.
    b. Click to select a stage to associate with the evaluation.
    c. Click *Yes* or *No* to specify whether notifications of policy violations will be sent as defined in the policy's configuration settings.
    d. Click the *Upload* button to begin evaluating the selected application. An *Evaluation Status* message is displayed.
6. When the evaluation is complete, click the *View Report* button to open the Application Composition Report (see page 193) for your application.



Policy Violation Comparison Behavior

Overview

We have changed the policy violation comparison (diffing) feature to make it more accurately highlight risk. An easily understood example is with security vulnerabilities.

> Currently: If a component has two security vulnerabilities with the same score it will be represented as a single policy violation.
>
> Change: We want to create two policy violations, which more accurately represents the risk this component poses.

This rationale also applies to policy configuration changes that are significant, since they represent a different and new understanding of risk.

Currently (July 2018), the policy violations are compared by policy internal id, policy name, policy threat level, component hash, and a component identifier. That is not enough for policy violations that are different based on the data that triggers the policy violation. We need to store extra data and use it in the policy violation comparison to achieve this new functionality.

With the new comparison model, some of the policy violation behavior will change. The purpose of this document is to illustrate the new behavior.

Waiver

Waivers will work the same way as before, based on the violating policy and component.

Notification

Recall that notifications are only generated for new policy violations, none are created when the policy violation is the same.

Old Behavior

A component has an existing Security-Medium policy violation, at policy evaluation a new security vulnerability appeared that meets the Security-Medium policy conditions. No notification is created since the policy violation is determined to be the same as the previous one.

New Behavior

A component has an existing Security-Medium policy violation, at policy evaluation a new security vulnerability appeared that meets the Security-Medium policy conditions. A notification is created since the policy violation is determined to be different due to the new security vulnerability.

Policy Rename

Old Behavior

Once a policy is renamed, any existing policy violations for that policy would be considered new when the next evaluation occurs. We used the rename as a shortcut for someone to indicate the policy is different after they changed conditions/constraints.

For example:

Evaluation on May 1, a policy named P1 has policy violation A.

Evaluation on May 2, a policy named P1 has policy violation A. The policy violation is considered old.

The policy named P1 is changed to be named P2.

Evaluation on May 3, a policy named P2 has policy violation A. The policy violation is considered new.

New Behavior

The policy name changes will not trigger new policy violations. The name is not used in determining changes we've introduced other heuristics to determine significant changes. Once a policy is renamed, any existing policy violations for that policy will remain unchanged when a new evaluation occurs (provided no other information changes).

For example:

Evaluation on May 1, a policy named P1 has policy violation A.

Evaluation on May 2, a policy named P1 has policy violation A. The policy violation is considered old.

The policy named P1 is changed to be named P2.

Evaluation on May 3, a policy named P2 has policy violation A. The policy violation is considered old.

Violation Data

The majority of the change is in how differences in policy violations are determined based on the data that triggers the policy violation (the conditions to be met). An explanation of the kinds of information and specific data fields used in this comparison follows, along with several examples.

Trigger Data by Condition Type

Label

- label id in the policy condition

License

- license id in the policy condition

License Status

- license status id in the policy condition

License Threat Group

- license threat group id in the policy condition

License Threat Group Level

- license threat group id
- the threat level of the corresponding license threat group

Security Vulnerability Severity

- security vulnerability reference id
- severity (score) of the corresponding security vulnerability

Security Vulnerability Status

- security vulnerability reference id
- the status id of the corresponding security vulnerability

Some Examples of Behavior

Label

- Old Behavior
    - A component has two labels: label-1 and label-2.
    - Evaluation with condition “Label is label-1” has a policy violation.
    - Change condition to “Label is label-2”.

- Evaluation with condition "Label is label-2" has a policy violation. The policy violation is considered old.

- New Behavior

- A component has two labels: label-1 and label-2.
- Evaluation with condition “Label is label-1” has a policy violation.
- Change condition to “Label is label-2”.
- Evaluation with condition "Label is label-2" has a policy violation. The policy violation is considered new because the label id has changed.

License

- Old Behavior
  - A component has license LGPL-3.0 and Apache-2.0.
  - Evaluation with condition "License is Apache-2.0" has a policy violation.
  - Change condition to "License is LGPL-3.0".
  - Evaluation with condition "License is LGPL-3.0" has a policy violation. The policy violation is considered old.

- New Behavior
  - A component has license LGPL-3.0 and Apache-2.0.
  - Evaluation with condition "License is Apache-2.0". A policy violation is created.
  - Change condition to "License is LGPL-3.0".
  - Evaluation with condition "License is LGPL-3.0" has a policy violation. It is considered new because the license id has changed.

License Status

- Old Behavior
  - A component has a license with a status of open.
  - Evaluation with a condition "License Status is Open" has a policy violation.
  - Change component license status to overridden.
  - Change condition to "License Status is Overridden".
  - Evaluation with a condition "License Status is Overridden" has a policy violation. The policy violation is considered old.

- New Behavior
  - A component has a license with a status of open.
  - Evaluation with a condition "License Status is Open" has a policy violation.
  - Change component license status to overridden.
  - Change condition to "License Status is Overridden".
  - Evaluation with a condition "License Status is Overridden" has a policy violation. It is considered new.

License Threat Group

- Old Behavior
  - A component has a license in the Liberal, and Copyleft license threat groups.
  - Evaluation with the condition "License Threat Group is Liberal" has a policy violation.
  - Change condition to "License Threat Group is Copyleft".
  - Evaluation with the condition "License Threat Group is Copyleft" has a policy violation. The policy violation is considered old.

- New Behavior
  - A component has a license in the Liberal, and Copyleft license threat groups.
  - Evaluation with the condition "License Threat Group is Liberal" has a policy violation.
  - Change condition to "License Threat Group is Copyleft".
  - Evaluation with the condition "License Threat Group is Copyleft" has a policy violation. The policy violation is considered new because the license threat group id has changed.

License Threat Group Level

- Old Behavior
  - A component has a license in a license threat group with a threat level of 4.
  - Evaluation with the condition "License Threat Group greater than or equals to 2" has a policy violation.
  - The components license has changed to a license threat group with a threat level of 5.
  - Evaluation with the condition "License Threat Group greater than or equals to 2" has a policy violation. The policy violation is considered old.

- New Behavior
  - A component has a license in a license threat group with a threat level of 4.
  - Evaluation with the condition "License Threat Group greater than or equals to 2" has a policy violation.
  - The components license has changed to a license threat group with a threat level of 5.
  - Evaluation with the condition "License Threat Group greater than or equals to 2" has a policy violation. The policy violation is considered new because the license threat group level has changed.

Security Vulnerability Severity

- Old Behavior

  - A component has a security vulnerability with a severity of 5.
  - Evaluation with the condition "Security Vulnerability Severity greater than or equals 5" has a policy violation.
  - The components security vulnerability severity changes to 6.

- Evaluation with the condition "Security Vulnerability Severity greater than or equals 5" has a policy violation. The policy violation is considered old.

- New Behavior

- A component has a security vulnerability with a severity of 5.
- Evaluation with the condition "Security Vulnerability Severity greater than or equals 5" has a policy violation.
- The components security vulnerability severity changes to 6.
- Evaluation with the condition "Security Vulnerability Severity greater than or equals 5" has a policy violation. The policy violation is considered new because the security vulnerability severity has changed.

Security Vulnerability Status

- Old Behavior

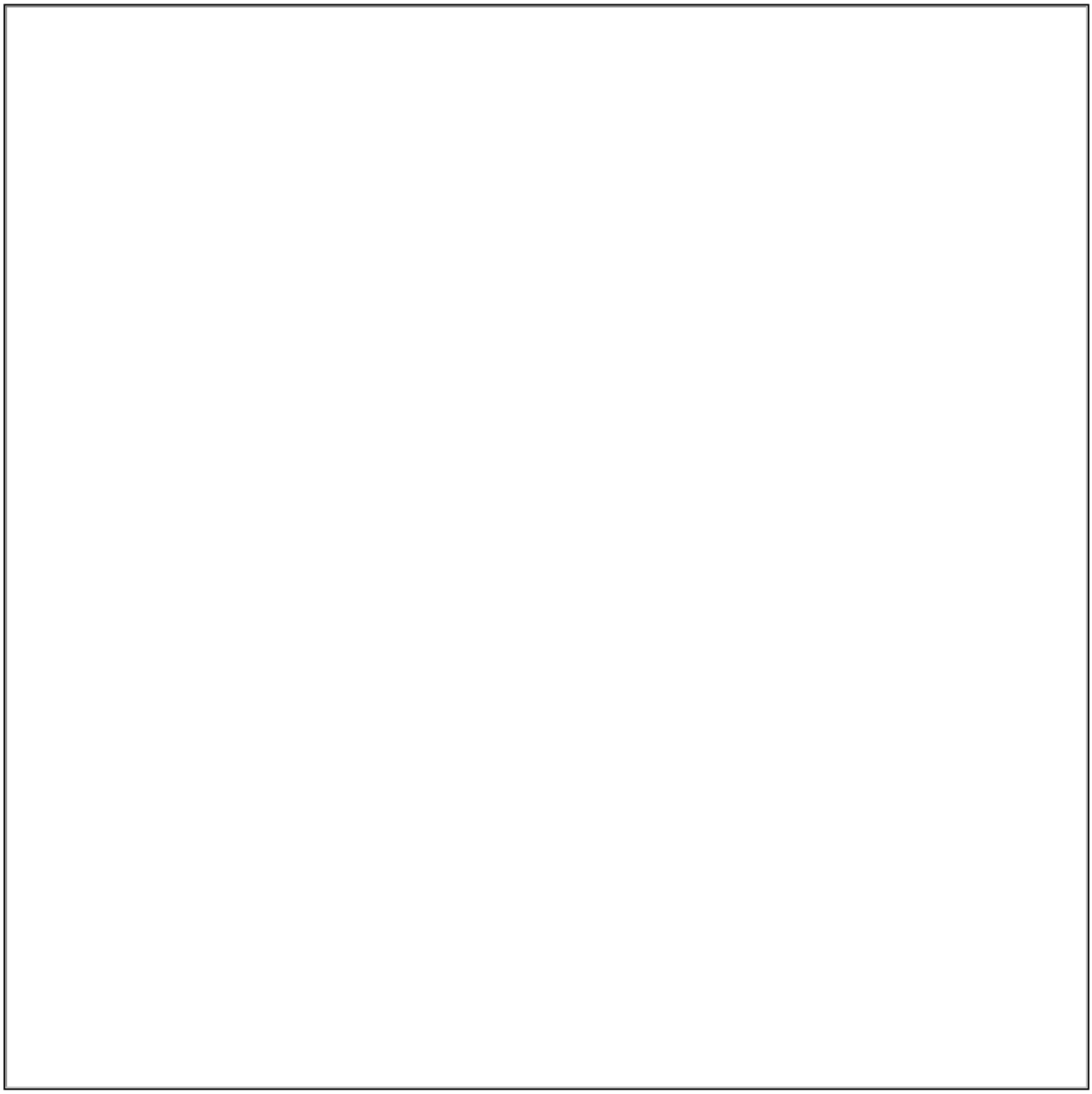
- A component has a security vulnerability status of open.
- Evaluation with the condition "Security Vulnerability Status is Open" has a policy violation.
- Change component security vulnerability status to acknowledged.
- Change condition to "Security Vulnerability Status is Acknowledged".
- Evaluation with the condition "Security Vulnerability Status is Acknowledged" has a policy violation but it is considered old.

- New Behavior
  - A component has a security vulnerability status of open.
  - Evaluation with the condition "Security Vulnerability Status is Open" has a policy violation.
  - Change component security vulnerability status to acknowledged.
  - Change condition to "Security Vulnerability Status is Acknowledged".
  - Evaluation with condition "Security Vulnerability Status is Acknowledged" has a policy violation is considered new.

Sample Application example

The sample application can be used to illustrate the changes to violation counts by reviewing the security policy violations.

Current behavior will ignore subsequent policy violations for the same policy. It appears as if the security vulnerabilities are grouped together as one policy violation per threat grouping. When looking at all the policy violations, notice that under Security-Medium the component names appear one time, indicating a single policy violation for the Security-Medium policy. When you view the details of the component you'll notice there are several security vulnerabilities.



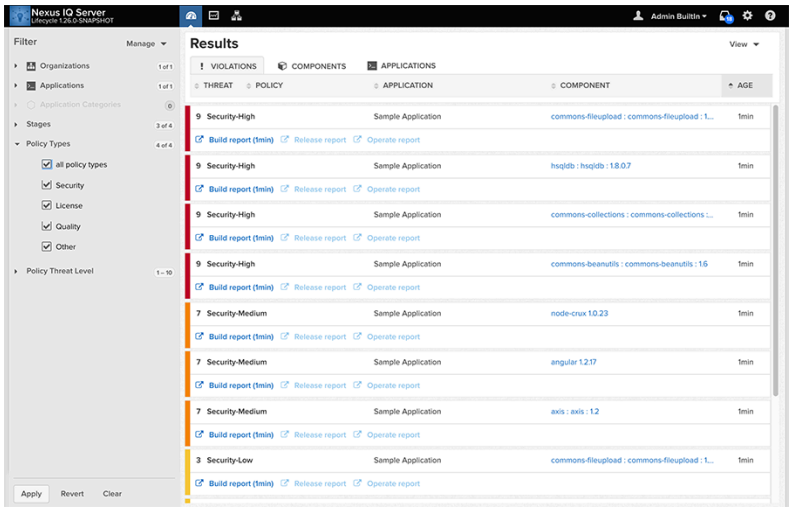
The new behavior will evaluate each security vulnerability and generate a separate policy violation for each. When looking at the policy violations, notice that under Security-Medium the following component names appear more than once because there is more than one security vulnerability that meets the policies conditions (security score within the medium range): geronimo, jetty, and tomcat-util.

Dashboard

The Dashboard provides the quickest way to review the overall health of applications you manage. The Dashboard is displayed by default when you log into IQ Server. If you are in any other location of the IQ Server, click the Dashboard icon on the IQ Server toolbar.

> Using the Dashboard requires IQ Server with the Lifecycle or Auditor license. The Dashboard only displays information for applications you are permitted to see, requiring a user assigned to the Developer role for at least one application.

The Dashboard is organized into two areas: Filters (see page 190) and Results (see page 190).



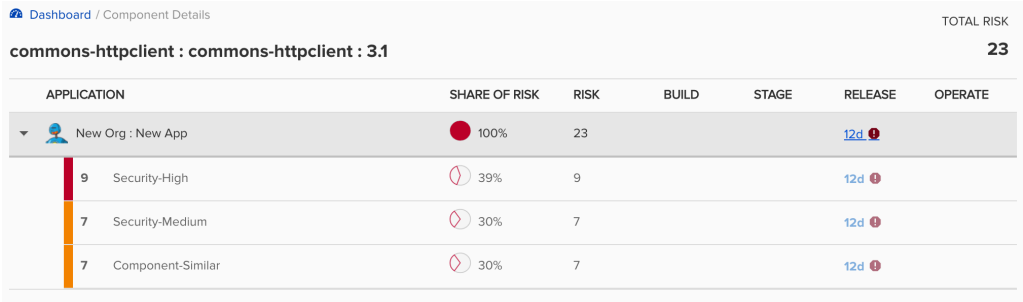
Filters

The *Filter* menu is located on the left side of the Dashboard. To adjust any of the various filters, click the filter label to see available options. For example, you can filter by organization, violations found within a specific stage, or policy type (see page 151). Once you've adjusted the filters, click the *Apply* button to update the violations list. To save an applied filter selection, use the *Manage Filters* menu.

Results

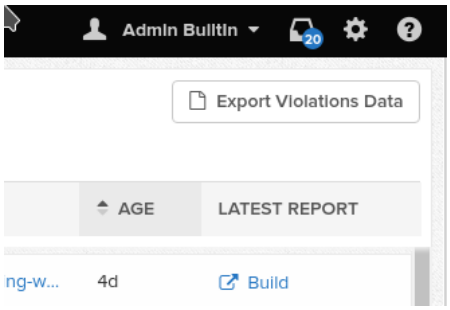
Dashboard Results display information based on applied filters. *Violations* is the default view for the Dashboard. It displays data for the last 30 days and shows the first one hundred, newest component violations found in your applications. You can also view *Component* and *Application* results using the tabs to change views.

Clicking a component in either the *Violations* or *Components* view opens the *Component Details* page. The *Component Details* page presents known coordinates for the component and all violations that have been found, organized by application. Risk information for each component is also provided:



Export Data

Export data displayed in the current results view using the *Export Data* button.



Results are saved on your local computer as a .csv file. The first row of the .csv file contains column names. The 'Date First Seen' column is in ISO 8601[103] format. The 'Timestamp First Seen' column has the same date, but in standard unix time[104] format.

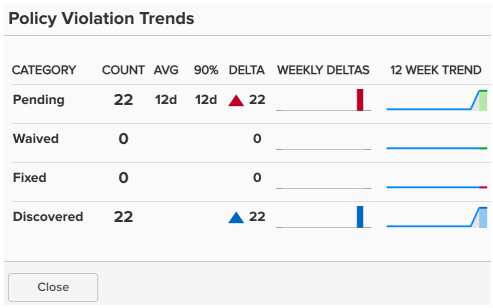
Policy Violation Trends

> ⚠ The *Policy Violation Trends* feature has been superseded by Success Metrics (see page 230), and was removed from IQ Server in version 1.40.

Prior to IQ Server 1.40, *Export Violations Data* was contained within a *View* menu alongside *Calculate Trends* . Selecting *Calculate Trends* from the *View* menu opens the *Policy Violation Trends* dialog. This shows policy violations trends for your current filter. *Policy Violation Trends* display a twelve-week look at how risk is entering your applications and how you are handling that risk, and also shows progress for all time. Calculating trends can take some time depending on the number and size of matching evaluations.

---

103 https://en.wikipedia.org/wiki/ISO_8601
104 https://en.wikipedia.org/wiki/Unix_time

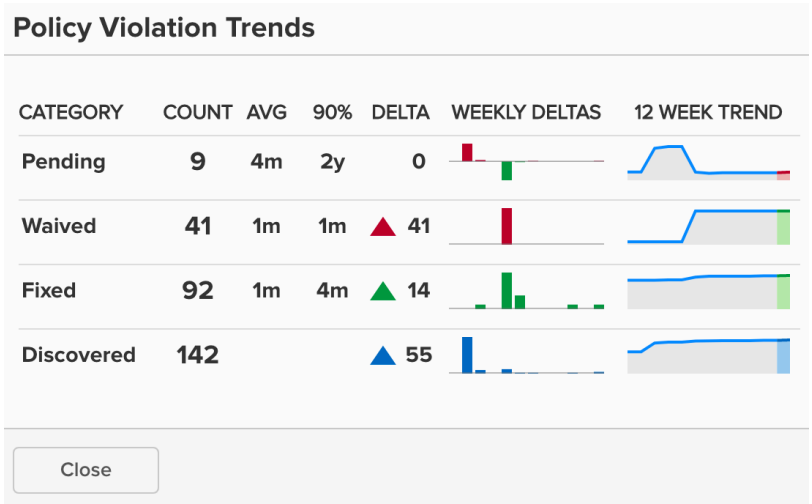


Interpreting Policy Violation Trends

> ⚠ The Policy Violation Trends feature has been superceded by Success Metrics (see page 230), and was removed from IQ Server in version 1.40.

At the top of the Results area, the View menu contains a *Calculate Trends* command. Calculate Trends opens a *Policy Violation Trends* dialog displaying policy violations trends that match your current filter.

The purpose of Policy Violation Trends is to provide a quick, twelve-week look at how risk is entering your applications, and how you are handling that risk. The information is divided into four categories, with each category displaying metrics over a twelve-week period.



So how do we interpret the example above?

A policy violation is *Pending* when it has been discovered, but not yet fixed or waived. Reducing the number of pending violations is a critical task. Weekly deltas above the x-axis indicate there were more discovered violations than those fixed; green bars below the x-axis represent more violations were fixed than discovered.

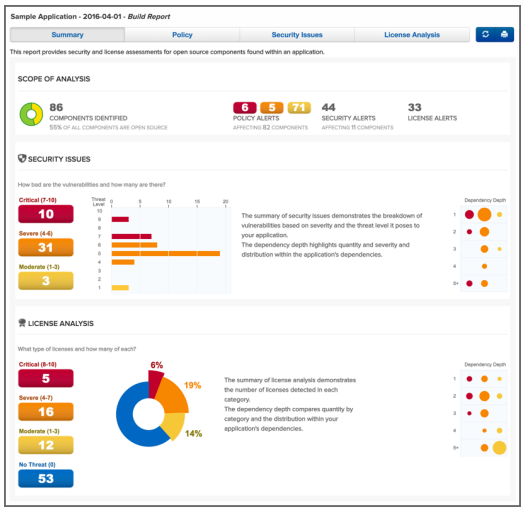
A policy violation is *Waived* when a particular component, either in the scope of this application or all applications for the organization, is waived from this particular policy.

A policy violation is *Fixed* when it no longer exists in any stage. When determining the fixed state of a component, any filtered stages are not considered. That is, if you exclude a stage where a violation has occurred, the count for fixed may increase even though the violation is still present in the other stage.

A policy violation is considered *Discovered* when it has been observed for the first time. It is not uncommon to see discovered violations trend upwards steeply, especially in the early phases of your implementation, and then plateau as you start developing a better component consumption process.

Application Composition Report

The Application Composition Report represents the health of your application. Ultimately, it serves as a point-in-time report representing risk associated with component usage for a specific application. The report includes information on how the application complies with the policies your team, or business, has established. In many ways, it's the final connector between policies and the components of your application.



When looking at the report the first time, it can be daunting. If you see tons of red, you may quickly be dismayed. Or perhaps, you don't see enough red and are worried in a different way. These feelings aren't uncommon, and they reveal another important aspect of the Application Composition Report - it contains a lot of information.

More than just reporting the violations components in your application have triggered, it also provides a way to improve policy management. These reports don't show false positives… ever. If there is a red, severe policy violation that should really be much lower, communicate back with the team in charge of managing the policies. In fact, of all its uses, the ability to communicate findings to a wide audience is perhaps the most important task of this report.

Related topics

- Accessing the Report (see page 194)
- Reviewing a Report (see page 196)
- Printing and Re-evaluating (see page 201)
- The Component Information Panel (see page 202)
- Resolving Security Issues (see page 204)
- Component License Information (see page 209)
- Component Identification (see page 212)
- Assigning Component Labels (see page 217)
- Waivers (see page 219)
- Policy Reevaluation (see page 223)
- PDF Report (see page 224)
- Impact of Improved JavaScript Reporting (see page 227)

Accessing the Report

Access the Application Composition Report from either the Reports Area or from the Organizations and Applications area.

Reports Area

Log into the IQ Server and click the Reporting icon  . If multiple applications have been scanned, you will see all of them here.

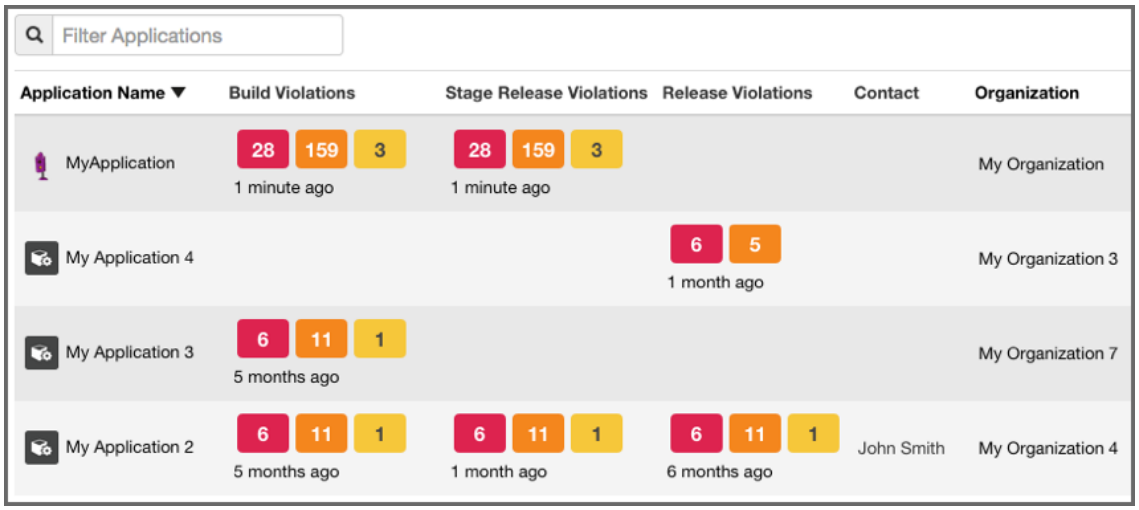
> You will need to be a member of at least the developer group for the application you wish to see a report for.

The Reporting screen shows several columns:

Application Name	Links to the Application Management Area for the specific application.
Build, Stage Release, and Release Violations	These three columns display the violation counts for the most recent evaluations. The counts are broken down by Critical, Severe, and Moderate with text indicating the time (e.g. 2 minutes ago) of the most recent evaluation. Clicking on a violation count opens the Application Composition Report for that stage.
Contact	The contact (if applicable) for the corresponding application.

Organization	Links to the parent organization for the corresponding application.

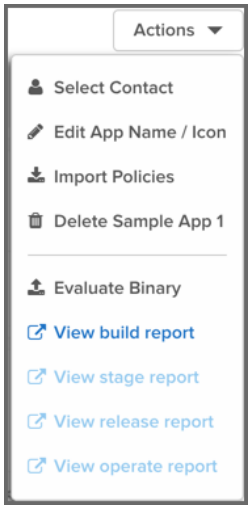
To access the Application Composition Report, click the violation count for the corresponding application and stage (build, stage release, or release).



By default this view will be sorted alphabetically by the application name. In addition to the filter, you can also click on the application or organization columns to sort alphabetically ascending/ descending.

Via the Organization & Policies Area

Select Organization & Policies  from the toolbar and select an application. You can access an Application Composition Report for a selected application from the *Actions* menu as shown in the figure below. The most recent report is available for the different stage(s) at which the application has been evaluated:



If you use the Nexus IQ CLI and don't specify a stage, it will default to Build. When your evaluation completes and the report is uploaded, it is accessible using the *View build report* action on the *Actions* menu.

Reports can also be accessed via enforcement point tools for CI and the repository manager. However, in each of the tools, they will connect to the IQ Server.

Reviewing a Report

Overview

The Application Composition Report is organized into four tabs: Summary, Policy, Security Issues, and License Analysis. These tabs represent the basic navigation for the report and serve to divide information into specific sections.

It's important to first understand a little bit about what a report represents and the basic sets of data it contains. In general, each report:

- Corresponds to a single, specific application, indicating the application name, date of the report, and the stage the scan took place in.
- Includes components found during a scan of the application, in most cases, including any dependencies.

- Records violations linked to an application's policies, or the policies inherited from the application's organization.
- Displays available security information for any components found matching components in the Central Repository.
- Displays available license information for any components found to exactly, or partially, match components in the Central Repository, as well as any data recorded manually (e.g. through the claiming process).
- Distinguishes between, external, proprietary and internally identified/claimed components.

Now that you know what forms the basis of the report, let's take a look at each tab individually.sc

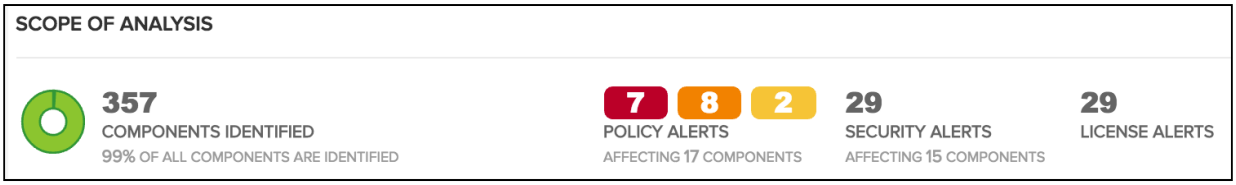
Summary Tab

The Summary tab is always the first section of the report displayed. It is broken into three sections:

Scope of Analysis

This section shows counts, giving you an idea of the volume of components that were found during the scan. It also gives a breakdown of those that were identified, including a specific percentage that is represented by open source components. In addition to these numbers, you will also see:

- A distinct count of components identified.
- A count of components with policy violations, displayed by threat level. Only the most severe violation for each component is counted.
- The total number of security alerts found, and the number of affected components.
- The total number of license alerts. Each license alert corresponds to a single component.



Security Issues

The Security Issues section provides three visualizations. The first visualization displays the number of security issues by their particular Common Vulnerability Scoring System (CVSS) score, breaking the issues into three threat levels - Critical, Severe and Moderate.

Next to this raw count, the same numbers are represented in a bar graph to help distinguish the relative impact for each threat level.

Finally, a dependency depth chart shows where the security issues occur, relative to how many there are, indicated by the size of the circles, as well as what level of dependency they are found in.

License Analysis

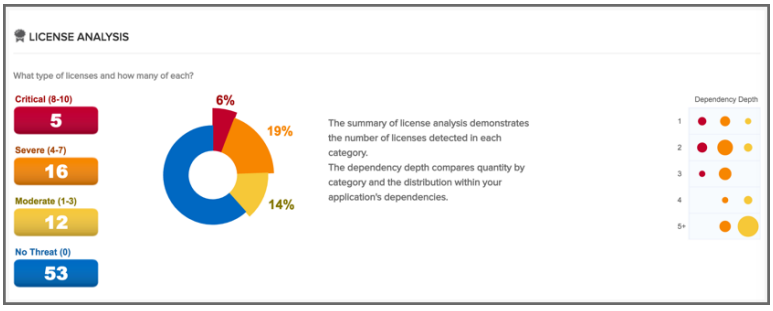
As with Security, the License Analysis section breaks the data into four threat level categories. However, these threat levels do not come from an external source, but rather the user-configurable license threat groups that are managed via the IQ Server.

There are four threat level categories:

- Critical (Copyleft)
- Severe (Non Standard)
- Moderate (Weak Copyleft)
- No Threat (Liberal)

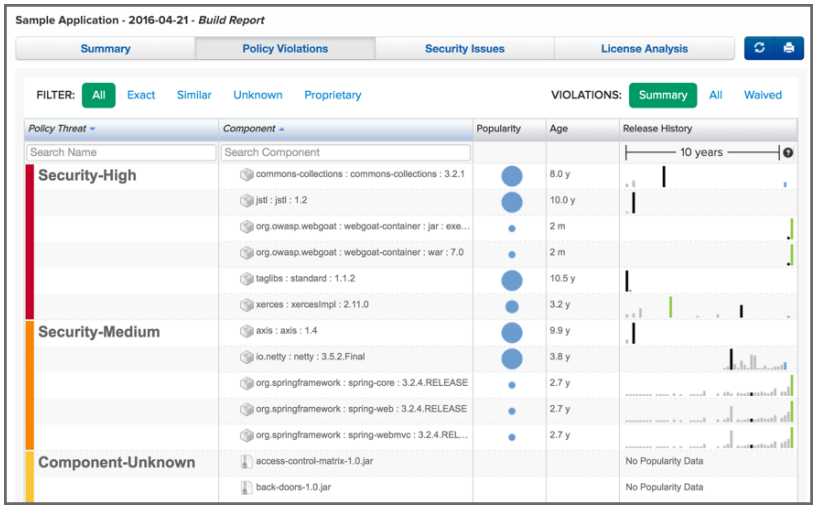
These categories are static and not not configurable.

The first counts that are displayed represent the total number of licenses found in each threat level. Next to this list, a graph indicates percentage of licenses in each threat level category, compared to the total number of licenses found. Finally, a dependency depth chart indicates the volume of licenses found at each dependency level, as well as the color corresponding to the threat level.



Policy Violations Tab

The Policy Violations tab displays a list of all components found during the scan of the application. By default components are ordered by their worst policy violation. This is an important distinction, because a component may have more than one violation, and the threat level severity for those violations could vary. If you wish to see all violations there are two options, using the Violation Filter, or the Component Information Panel (CIP). Below we have highlighted the available filters.



Filter

The filter lists five categories:

- All (default)
- Exact
- Similar
- Unknown
- Proprietary

In addition to the main set of filters, you can also filter by violations, including those that have been waived. The available options include:

- Summary (default)
- All
- Waived

Clicking on any of these will change the components in the list. We'll discuss each of these in further detail in the sections corresponding to component matching, claiming components, and waiving components sections.

Component List

The list of components, below the filter, displays the threat level posed by the components. The *Policy Threat* column displays the name of the worst violated policy for the component and the severity using a colored bar. The *Component* column displays all available coordinate information for the component.

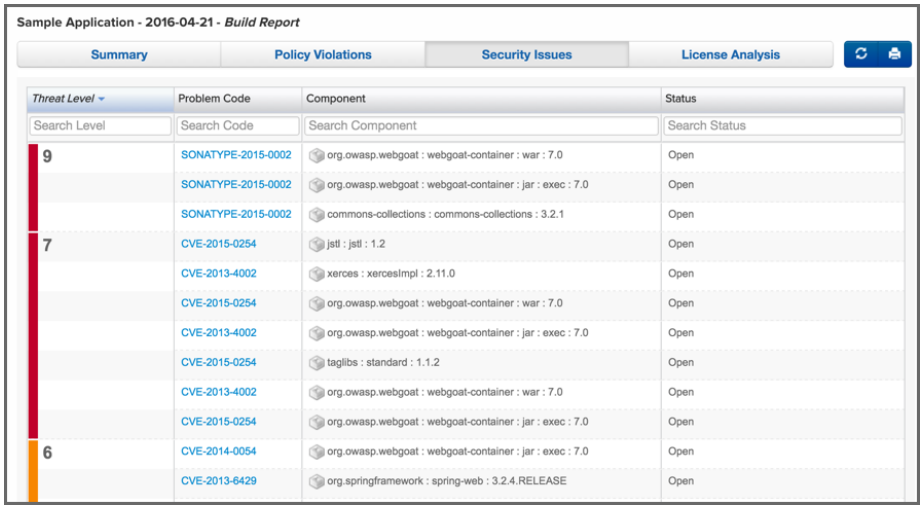
In addition the list displays the *Popularity* and the *Age* of the component in the Central Repository in separate columns. The *Release History* is displayed in a visualization that includes the most popular version, the most recent version, your version and any other available versions in a timeline.

By clicking on the column header, the list of components can be sorted. If you are looking for a specific policy, or component, you can use the search fields located at the top of each of those columns, directly below the header.

Clicking on a row for a component in list displays the Component Information Panel (CIP). For more information on the CIP, see The Component Information Panel (see page 202).

Security Issues Tab

The important thing to remember about the Security Issues tab is that information displayed there is related specifically to security vulnerabilities data that has been collected by Sonatype. This data however, is separate from policy violations, which are based on policies that you have created (or imported), and are displayed on the *Policy Violations* tab. That is, you could certainly have a situation where there is a security vulnerability, and no policy violation. Because of this, it is important to treat them independently.



The way components are displayed is actually quite different as well. In the Security Issues tab, only those components with a security vulnerability are displayed. The data provided for each component is broken into several columns:

- Threat Level
- Problem Code
- Component
- Status

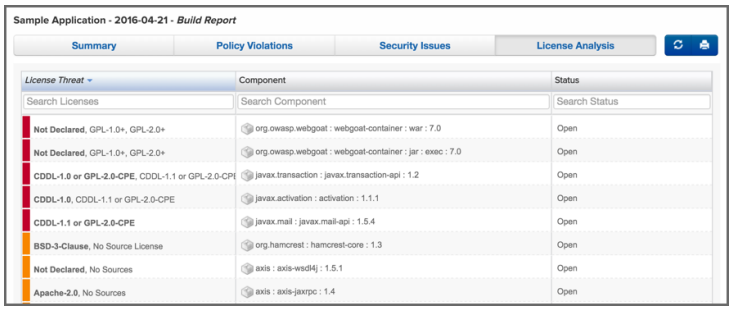
By default the list of components with security vulnerabilities is organized by threat level. This helps you isolate the most critical issues you need to address. However, you may notice that components in this list are repeated. This is because a component may have more than one security vulnerability, and those vulnerabilities in fact may have different scores, thus different threat levels.

To sort the list, simply click the corresponding header. For example, if we wanted to sort by components, finding a component with multiple vulnerabilities, we would simply click on the *Components* column.

Additionally, you can search for a specific component by typing in the search field located directly below each header.

License Analysis Tab

The License Analysis tab displays all identified components found in the application scan and their license threat details. Unknown components are not displayed. Similar to the security issues, a license threat does not necessarily correlate to a policy, and as such should be treated independently.



For each component listed, the license related data is displayed. This data is based on information collected during a scan. By default, components are listed based on the threat of the corresponding License Threat Group that identified license is in. However, like the other tabs, clicking on a column in list will sort the components by that column. Additionally, specific components can be isolated using the search located below each header. The columns displayed include:

- License Threat
- Component
- Status

Printing and Re-evaluating

The top right corner of the report displays two buttons that give you access to refreshing the report as well as printing the report:

The refresh button on the left triggers a re-evaluation of the report. It will take the existing list of components in the report and reevaluate them against the your application policy.This comes in handy when you are making policy changes and want to see how that would affect the current data without having a to rerun a build.

The second icon on the right, the printer icon, allows you to create a PDF version of the report that is nearly identical to the HTML version. You can use this report for actual printing on paper, distribution to recipients

without access to the IQ Server or simply for archival purposes. And of course it also works as a great bill of materials for your application.

The Component Information Panel

Overview

Clicking on a specific component opens the Component Information Panel (CIP). The first thing you should notice is that the CIP can be accessed for a component on the *Policy*, *Security Issues*, and *License Analysis* tabs. No matter which of these tabs you are on, simply click on the component, and the panel is displayed. Even better, the information displayed is the same, regardless of the tab in which you clicked on the component.

The CIP itself is divided into two areas. The top has a list of various sections, each providing more specific details and functionality related to the component. Below these sections, the panel will display information for the corresponding section. A brief description of each section is included below.

Component Info

The following table outlines the details provided for a chosen component in the Component Info tab:

Declared License	Any license that has been declared by the author.
Observed License	Any license(s) found during the scan of the component's source code.
Effective License	Either any licenses included in the Declared or Observed Group, or the overridden license.
Coordinates	The identifying information for a component. For known components, all available coordinate information will be displayed.
Highest Policy Threat	The highest threat level policy that has been violated, as well as the total number of violations.
Highest CVSS Score	The highest threat level security vulnerability and the total number of security vulnerabilities.
Cataloged	The age of the component based on when it was first added into the source from which it was identified.
Match State	How the component was matched (exact, similar, or unknown).

Identification Source	Whether a component is identified by Sonatype, or claimed during your own process.
Website	If available, an information icon providing a link to the project is displayed.

The graph itself is laid out like a grid, with each vertical slice representing a particular version. The current version is identified by a vertical line. The information displayed in the graph includes:

Popularity	The popularity for each version is shown as a bar graph. The taller the bar, the more popular the version.
Policy Threat	The heatmap marker colors represent the highest policy threat levels for each version across all policy types (see page 151), with no marker indicating no threat.

Policy

The Policy tab shows details of any policy violations for the component. Here you can see the name of the policy that has been violated (and any action that was taken), the name of the constraint that has been violated, and the value that was found. While the *Policy/Action* and *Constraint* names are straight forward, the *Condition Value* may be a little confusing at first.

A condition is simply the *if* part of an *if/then* statement. If a certain condition value is found which is equivalent to a condition being met, then the policy will be violated. E.g. if we have a policy that has a condition such that if a security vulnerability is found, our *Condition Value* column would indicate, *Found x Security Vulnerabilities*. In the same regard, *Constraints* are simply multiple conditions joined together.

In addition to simply viewing the policy information details, a policy violation can also be waived (see page 219) in this section of the CIP using the *Waive* button.

Similar

Any components found to be similar to the selected component will be listed in the *Similar* tab. For example, a similar component could be a component that a developer has built locally using the source code of an open source component with minor modifications or additions.

Occurences

When a file is scanned, it has a filename and location where it was found. In some cases, it may have more than one filename and location. Either way, the path to the location(s), as well as the filename(s), of the component that was scanned is included in this section. In short, the Occurrences section lists the file names and locations where the component was encountered. This section can be especially useful to detect accidental shipping of duplicate components archives or a misconfiguration of your actual report creation target e.g. you might be scanning the deployment archive (e.g. a war file) as well as the build output folder used to create the archive.

Licenses

The Licenses section is split into two areas. On the left, any licenses that were identified as declared by the author of the component, as well as any license found during the scan of the component source code are listed. On the right, is the license status area where you can set the *Status* of the component license information.

Vulnerabilities

The Vulnerabilities tab is separated into two areas. On the left, all security vulnerabilities related to the component are displayed. Clicking the **"i"** info button on any of the vulnerability rows will show more details. On the right is the security vulnerability status area.

Labels

The important item to note here, is that the assignment of labels to a component is done in this section of the CIP.

Claiming Components

The Claim Component tab is only available for unknown or similar component matches. During a scan, some components are identified as unknown or similar. Since we realize that in many cases you actually recognize these components, we provide this section to claim these components.

Audit Log

When changes are made to the status of a security vulnerability, or the status of a component's license within the scope of a particular application, that information is recorded in the Audit Log tab of the CIP for that component.

Resolving Security Issues

Evaluating your application for the first time, and seeing a huge number of critical security vulnerabilities indicated on the Summary tab can be a sobering experience, and in some ways it should be a little worrisome. More importantly though, it should create motivation for further investigation.

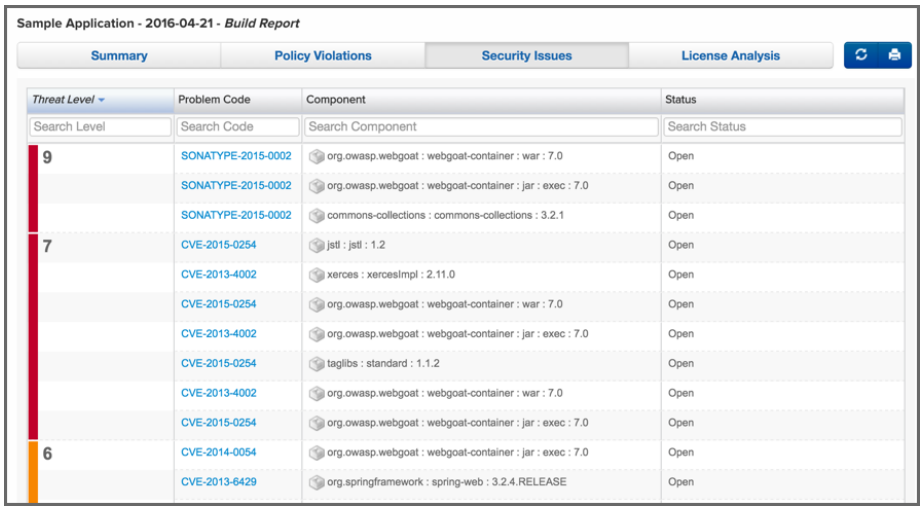
The key word there being investigation. That's because even though we've provided accurate data, you still need to have a process to review all available data, and then track your progress. It is not completely uncommon, and quite possible that a vulnerability doesn't apply to your application or, at the very least, isn't

a concern given the particular application you are developing, and it's relative exposure points. Where do you start your investigation though?

Security Issues

The component list on the *Security Issues* tab only shows components that have a security vulnerability. If a component has multiple security vulnerabilities it is displayed multiple times.

There are a total of four columns: *Threat Level*, *Problem Code*, *Component*, and *Status*. Initially the list of vulnerabilities is ordered by the *Threat Level* column. However, you can sort the list by any other column by simply clicking on a header.

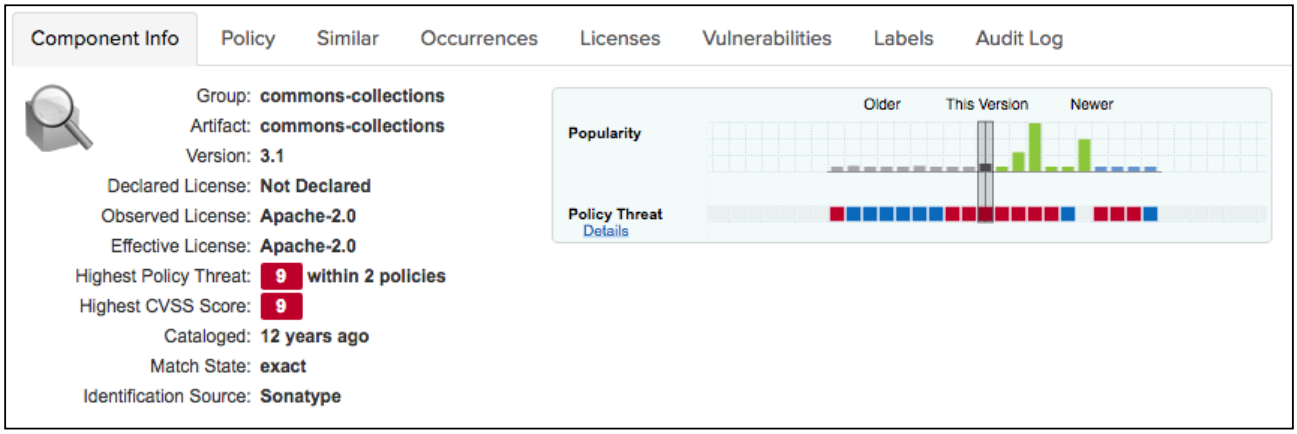


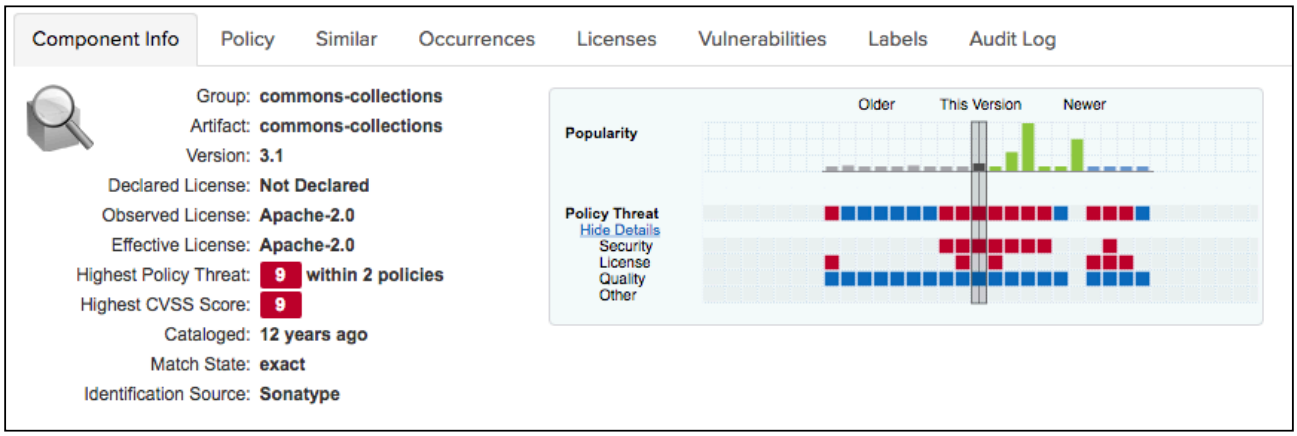
While the *Threat Level* and *Component* columns are self-explanatory, the *Problem Code* and *Status* columns deserve a bit more explanation:

- The *Problem Code* column provides a link to available details for the security vulnerability on the CVE and OSVDB web sites. This information is provided via the CVE and OSVDB security information sites. These public security databases allow you to get quick information about the security issue and nature of the vulnerability.
- The *Status* column tracks the state and progress of research of the effect of a security vulnerability with respect to your application. We'll focus on the Status column in a bit more detail when we cover the CIP. A key point to remember, is that as long as the status is set to Open, Acknowledged, or Confirmed, the vulnerability will be included in the counts on the summary page. In addition, a policy with a condition related to the presence of a security vulnerability will be met, as long as the status is set to Open. That means it's very important to research these issues, so that only those affecting your application remain.

The Component Information Panel (CIP)

To access the CIP, simply click on a component row in the list. There are three sections you should use during your security vulnerability investigation - Component Info, Vulnerabilities, and Audit Log.





Component Info

One of the first things you should notice in the *Component Info* section is the *Highest Policy Threat*. This field, located on the left side of the panel, displays the highest threat level policy that has been violated, as well as the total number of violations.

Next, you should take a close look at the graph to the right of the panel. On the heatmap, locate the row for Policy Threat. This graph will display the highest policy threat levels for each version across all policy types[105], with the current version identified as This Version. A breakdown of the highest policy threats for each policy type[106] can be displayed by clicking on the Details link. In some cases there are clear points

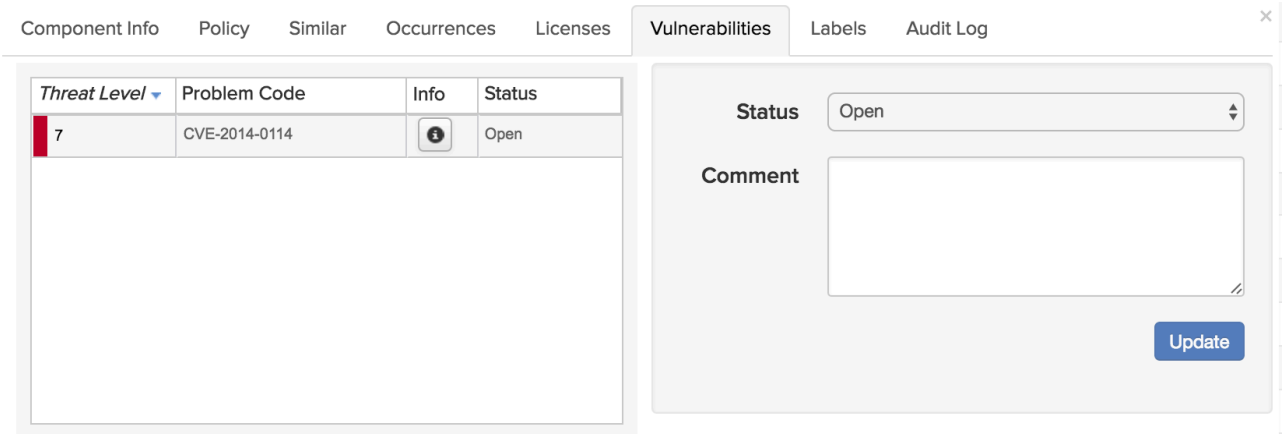
105 https://help.sonatype.com/display/NXIQM/Understanding+the+Parts+of+a+Policy#UnderstandingthePartsofaPolicy-SettingConditions

106 https://help.sonatype.com/display/NXIQM/Understanding+the+Parts+of+a+Policy#UnderstandingthePartsofaPolicy-SettingConditions

where Security policy threats have been resolved, as can be seen above. Often this tends to coincide with more popular version, although, that is not necessarily always the case.

Vulnerabilities

After clicking on a component row to display the CIP, click the *Vulnerabilities* tab.



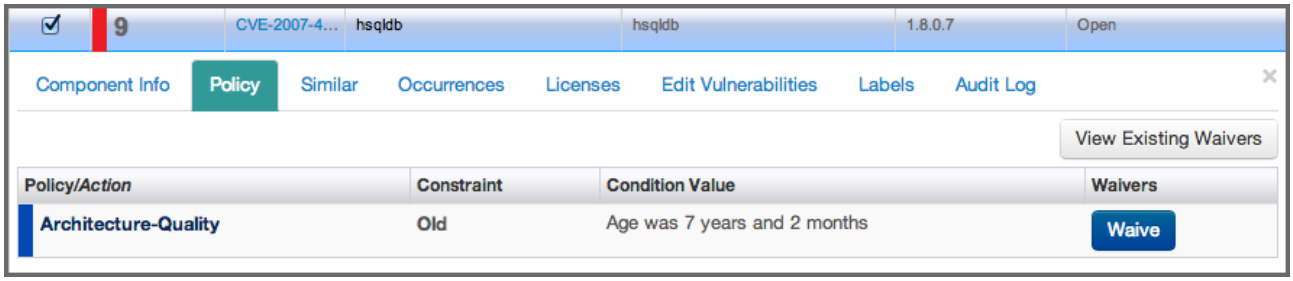
Here, the left side displays all security vulnerabilities. Depending on how many, this list may scroll. The list is organized into four columns:

Threat Level	Indicates the threat assigned to the security vulnerability and is determined based on the source. This is not associated to any policy threat level.
Problem Code	This is the unique identifier of the security issue as assigned by the source (e.g. CVE-2000-5518). It will change depending on the source of the data.
Info	Sonatype provides information from public sources, as well as information from our own research team. Clicking on the information icon in the corresponding row displays additional details provided about the issue.

Status	The status of the security issue as assigned by the drop down to the right. See below for information on changing this status. To the right of the list of security vulnerabilities is the status drop down and a comments section. To change the status simply select one from the drop down, select the vulnerabilities the status will apply to, enter any associated comments, and finally, click the Update button. After you click the Update button, any Status Comment you entered will be cleared, but is saved and visible in the Audit Log (see page 202). It is important to mention the status can be changed to any status at any time. There are four statuses available: • Open: The default status, represents no research being done. • Acknowledged: Represents that the security vulnerability is under review. • Not Applicable: Indicates that research was conducted, and the particular vulnerability does not affect the application. • Confirmed: Demonstrates research was conducted, and it has been determined the security vulnerability is valid and applicable. ⚠ Updating the status of the security issue requires the "Edit IQ Elements" permission.

Matching to Violations

In some cases, just because there is a security vulnerability, that does not necessarily mean there is a corresponding policy violation. For this reason, it's important to refer back to your *Policy Violations* tab as well. If you are finding that critical security issues you are troubleshooting do not show up as a policy violation as well, you may need to refine your policy so that future security issue trigger a policy violation and thus ensure that they get your attention.



Component License Information

In some cases, the licenses of a component is the last thing a development team will think about. This could simply be due to a misunderstanding of open source, or a situation where it's nearly impossible to do the exhaustive research needed to determine the license for a given component, especially dependencies.

Even if you haven't built policies around licenses the *License Analysis* tab provides license information about every component found in during a scan of your application.

This license information is provided via data collected from the Central Repository, as well as research conducted by Sonatype. In addition to the license information for each component, we'll also assess a threat of each license, based on a set of default License Threat Groups. As with Security Issues, the best place to start is with the component list in the *License Analysis* tab, and then move into looking at additional details for individual components, making any license status changes as you see fit.

License Threat Group

License threat groups are based on what is configured for each organization or application. Additional information can be found in License Threat Groups (see page 171).

How you manage your license threat groups directly impacts how threat is translated in the reports.

License Analysis

The component list on the *License Analysis* tab is more similar to the list on the *Policy Violations* tab, because it is a list of all components, not just those that have a license issue.

The list itself includes columns for *License Threat*, *Component*, and *Status* of the license issue. Clicking on the column provides sorting, while specific items can be searched using the field just below the column heading.

License Threat

The list of components is ordered by license threat which is based on the threats assigned to the license threat groups. Though a single component may actually have several licenses, license threat will only show the highest threat. This threat, as we mentioned earlier, is based on four default categories, which correspond to four default license threat groups of the same name.

- Critical
- Severe
- Moderate

- No Threat

Status

License status, like status for security vulnerabilities, allows you to track the process for license related research. In addition it provides a way to override a license in situation where you believe the license to be incorrect, or there is an option to choose a specific license.

The Component Information Panel (CIP)

To access the CIP for a component on the *License Analysis* tab, simply click on the component row. It will expand providing details in a number of sections. You will likely notice this looks the same as other CIP panels when clicking on other tabs of the Application Composition Report, and you would be correct. There is nothing additional provided by accessing the CIP via the *License Analysis* tab of the report. However, for this section, we want to focus on the license related information in the Component Info section, as well as the entire *Edit Licenses* and *Audit* sections.

Component Info

Again, the information contained here would be the same, whether or not you clicked on the component in the *License Analysis* tab. However, this gives us the context to talk about the *License* related fields in this section.

License Identification Types

On the left side of the Component Info section, you should pay attention to three fields, which are described below.

- Declared License: these are the licenses that the developer of the component has identified.
- Observed License: these are the licenses that have been observed during Sonatype's research.
- Effective License: the effective license displays license information based on one of two scenarios. In cases where multiple licenses are found, including any that are observed, these will all be included as effective. If a license is selected, or overridden, then that license will be considered effective, and listed here.

License Identification Values

In cases where there is no declared and/or observed licenses, a message will be displayed. There are several options, each with specific meaning:

- No Source License: sources were provided, but there was no license data found.
- No Sources: indicates we have no sources for the component.
- Not Declared: indicates nothing was declared by the author/developer.

- Not Provided: will appear when the license is actually null, and is unique to claimed components, but might also happen while new components are being processed by Sonatype.
- Not Supported: indicates Sonatype or the target ecosystem does not currently support automated license collection for this component format.

Component Graph

The graph itself is laid out like a grid, with each vertical piece representing a particular version. The selected version being identified by a vertical line.

While the information displayed in the graph includes information about popularity and violations of all types of policy, right now, just take a look at the policy threats for licensing (Click on the Details link to show policy threats for each policy type). The heatmap provides a quick way of identifying a component version with a suitable license.

Editing License Status and Information

Editing a license can be used for different purposes. One addresses the workflow of your research into a license related issue, while the other allows you to completely override a license all together. We'll cover all this below, but first let's take a look at the information displayed.

After clicking on a component in the list, and then the Licenses section of the CIP, the left side of the CIP displays the license(s) declared by the developer of the component, those that have been observed, and what is considered effective (a combination of the previous two). That is, unless they have been manually overridden or a specific license has been selected.

Next to each of these licenses is a box, displaying the severity of the license. This list can get long, so you may have to scroll to see all the licenses. Then, to the right of the license list, there are four drop down lists.

Scope

Scope allows you to apply the license status to this component by choosing application or to all components attached to the current application's organization by choosing organization.

Status

As we mentioned previously, Status provides a way to track your research, override a license, or select from an option. The available options are included below.

- Open: This is default status, and will be included in the count of license issues.
- Acknowledged: Acknowledged indicates the issue is being researched, and will still be included in the count of license issues.

- Overridden: This status will allow you to select one or more licenses from the License(s) dropdown (located just below the Status dropdown). This will override any licenses that have been declared or observed.
- Selected: In cases where there are multiple licenses, this option will populate the License(s) dropdown with any licenses found in the component, declared or observed. Multiple licenses can be selected.
- Confirmed: Confirmed simply indicates that the license(s) found are indeed correct, and will be included in any count of license issues.
- License(s): The License(s) drop down only displays given that a status of selected or overridden has been chosen. Given that it will present either a list of all licenses (if override is chosen) or only the declared and observed licenses (if selected is chosen). The license that is chosen will be displayed in the Effective License field in the Component Info section of the CIP. In addition, any overridden/ selected license will be indicated with a label of same name, next to the license in this field.

Comment

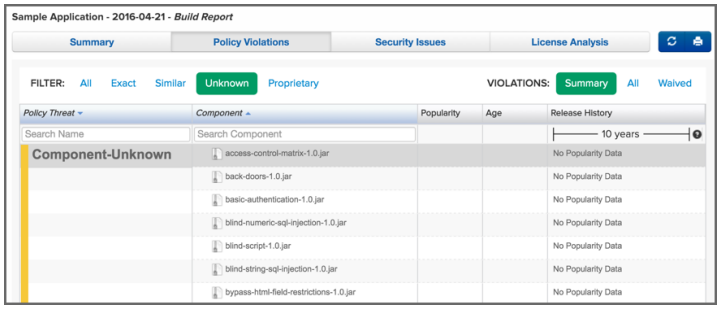
A comment is not required, but is a good element to include whenever you are making changes to the License Status. This is because it provides a way to understand, as well as audit, the decisions made to change a license status. This comment will be included with the record in the Audit Log section of the CIP.

Once you have made all your selections, and entered any necessary comments, click the **Update** button to save the License Status change.

Component Identification

Overview

One of the most important things you can do with regards to understanding the components in your application, is to identify them. What remains unidentified is of obvious concern.



Components can be identified in a number of ways, including:

- Extensive matching via various, proprietary algorithms
- Claiming components

- Establishing proprietary components

In this section, we'll describe all of these in detail, within the context of identifying components using the Application Composition Report, as well as offer our suggestion for best practices.

Matching Components

When an evaluation is performed, hashes of the components in your application are created. This in many ways is like a fingerprint, which is unique to a component. That fingerprint (hash), is compared back to components known to the IQ Server, which will provide all the available component info. This includes: usage statistics, security vulnerability, and license information.

All of this information can be used as parameters in your policy, which translates to more understanding of the component usage in your organization. That data however, can only be linked based on a matching of hashes, which can be exact or similar, and in some cases, unknown. We discuss these three match types in the table below.

Exact	An exact match means that a one-to-one link was found between a component hash in your application, and a component known to the IQ Server. This is the best case scenario with regard to component identification, and most components should fit in this category.
Similar	A similar match is found using various, proprietary matching algorithms. In a way it's a "best guess" to match a component that you have in your application with a similar one known to the IQ Server. In some cases, multiple matches may be found, and this is where the Similar section of the CIP is important. While the most likely match is used to display any information about a similar matched component, you can see all other matches in this section of the Application Composition Report.
Unknown	There are instances where not even a similar component match can be determined. This should be considered a serious situation, at least one that needs to be investigated. This could be a case of a component being recompiled and modified so that a match is no longer possible. However, there is a chance that component is something malicious introduced into the application. Either way, an unknown component should prompt an investigation. Of course, if during your investigation, you are able to identify the component, you can claim that component, via the Claim Components section, which we will walk you through in more detail a little bit later. ⓘ Unknown components will not be displayed in the *License* tab until they have been claimed.

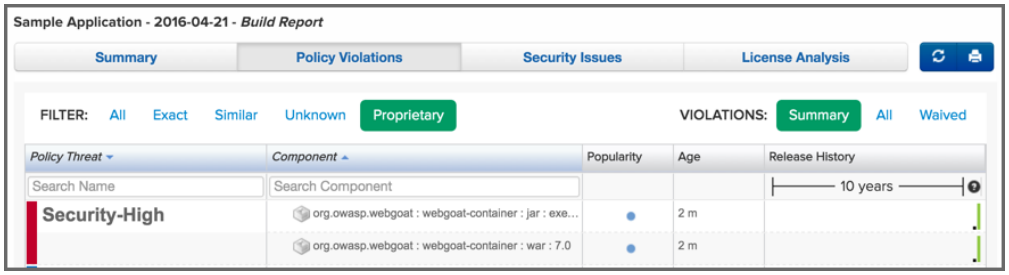
In addition to the main filters above, you can also control whether all violations for each component will be displayed. By default the summary of violations is shown. This means that only the worst violation for a component will be shown, and the component will only appear once in the list. Choosing *All* or *Waived*, will show all violations (including those waived), or only the waived violations, respectively.

> Changing the Violations filter can result in the components being displayed in the component list more than once.

Managing Proprietary Components

Proprietary components are unique to your organization. In many cases, these are developed by your organization and distributed among the applications you create.

In the Application Composition Report, you can view proprietary components on the *Policy Violations* tab as shown below. While *Proprietary* is listed under *Filter*, the process for matching exact, similar, and unknown components is separate from identifying a component as proprietary. You can have proprietary components that are an exact match, similar match, or unknown match. If you're trying to determine which of your components are proprietary, a good place to start is to review unknown components.



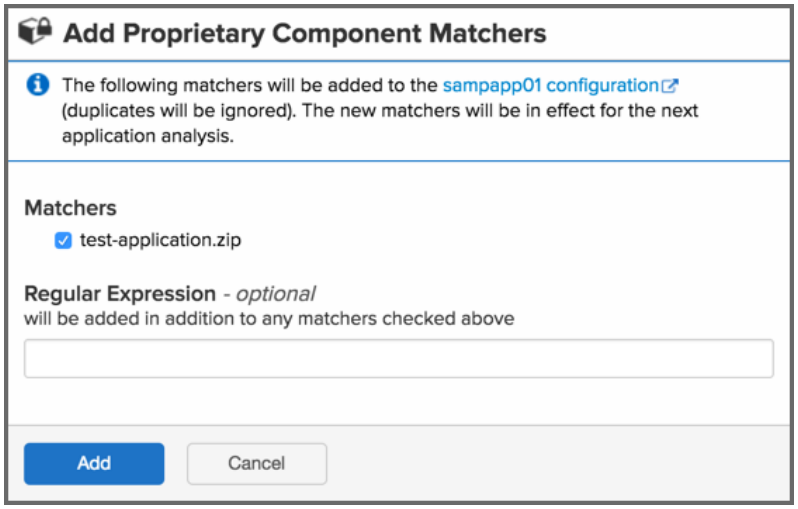
IQ Server uses *proprietary component matchers* to identify proprietary components when applications are evaluated. Proprietary component matchers are configured in the *Organization & Policies* area, but you can also add them from within the Application Composition Report. This is especially useful when the report contains unknown components that you know are proprietary.

> You need to be assigned to a role with "Edit Proprietary Components" permission in order to add proprietary component matchers. The built-in Policy Administrator and Owner roles have this permission.

To add proprietary component matchers:

1. Click to select a component.
2. Click the *Add Proprietary Component Matchers* button.
3. In the dialog, click the occurrences of the component that are proprietary.

4. Optionally, add a regular expression to identify proprietary components.
5. Click *Add*.
6. Re-evaluate the binary application or repository to see the component identified as proprietary.



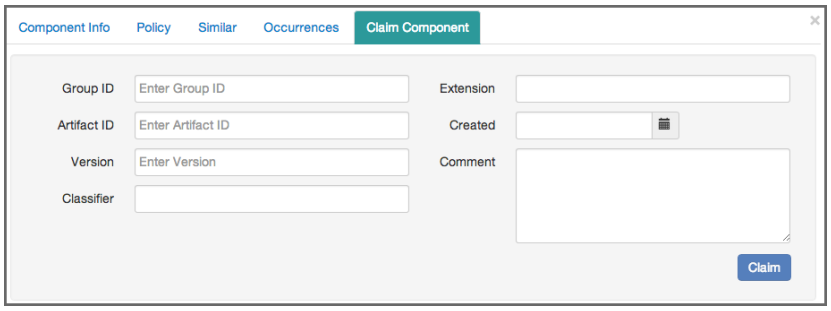
Existing reports are unaffected by additions to the Proprietary Component Matchers list; only after the next evaluation will you see the newly added components identified as proprietary.

For more information about the Proprietary Component Matchers, see Proprietary Component Configuration (see page 166).

Claiming a Component

When a component is *similar* or *unknown*, yet you are certain the component is recognized by your organization, you can prevent that component from being identified as similar or unknown in future reports. In other words, you can claim the component as your own.

Once claimed, that component will be known to the IQ Server. It will no longer be treated it as *similar* or *unknown*, and instead result in an *exact* match.

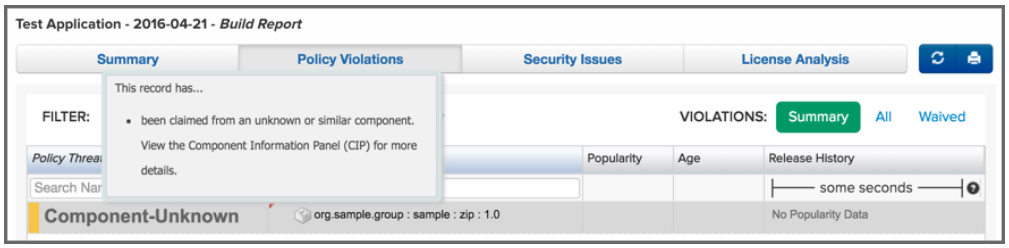


1. Access an Application Composition Report.
2. Click the *Policy Violations* tab, and then click the *Unknown* or *Similar* component filter.

3. Click the row of component you wish to claim in the list - the *Component Information Panel* is displayed.
4. Click on the *Claim Component* section of the CIP .
5. Enter values for the coordinates of the component.
6. As an option, enter the *coordinates classifier*, the *Created Date*, and/or a *Comment*. The created date is initialized with the date of the youngest entry in the component to be claimed.
7. Click the *Claim* button, to officially stake your claim for the component.

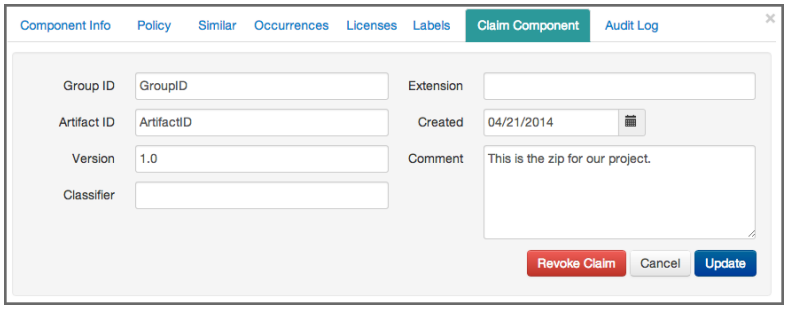
On review of the existing report, as well as those in the future, there is now an indicator that information about the component has been edited. When hovered over, a tooltip is displayed identifying that the component has been claimed.

We refer to this as the edited component tick mark (a small red triangle) on all future scans for this application, as well as any application with a valid Application ID on the IQ Server.



In addition, the Component Info section for the claimed component will now have two new fields, one indicating the *Identification Source* is *Manual*, and the other, *Identification Comment* will include any comments that were entered. While any policy violations will be displayed, the component graph will not.

Finally, if you have made a mistake and wish to revoke the claim on the component or make an edit, click on the *Claim Component* tab. Then, use the *Revoke* or *Update* buttons respectively.



Use the cancel button to undo any changes you made but haven't saved.

Assigning Component Labels

What are Component Labels?

Component labels are one of the more powerful features, though they operate similar to other label or tagging systems you have likely used. Basically you create a number of labels or tags as a set of available values and then assign them.

For example in a photo collection you could have labels for the content like *sunset*, *mountain*, *ocean*, *waterfall* and so on. An individual photo could then have multiple of these labels assigned. In a similar way you can use component labels to identify a particular type of component.

A common approach is to use component labels to identify an approved component or a component that needs research. Component labels can be anything you desire and can have a number of contexts as well. Some component labels might be architecture related, while others are related to legal and security properties and yet others are simply signaling ownership by a specific team. The flexibility of the component label system allows you to design your own use cases and implement them.

Component label creation and management is performed in the IQ Server *Organization & Policies Management* area. For more information on creating component labels, see Component Labels (see page 168). Assigning those labels to a particular component is a function of the Application Composition Report.

Where do Component Labels Begin?

Initially, component label assignment might seem like a task that deserves less process. It's actually the opposite.

The component label process actually starts at the IQ Server, where each organization, in many cases application, has a specific set of component labels.

Creating component labels (see page 168) at this level means that they are available to users of the Application Composition Report. So, before you label something, a number of things need to be considered, which might mean you need to go back to component label and policy management. Let's take a look at some questions that we can use to form a baseline for what component labels we need and/or should have.

Do we have a process defining component labels, as well as how and when they should be used?

> Every component label should have a reason for using it. If it doesn't you don't need it. That's because people will naturally infer meaning from the label. For example, if you have a needs review component label, you should have a process for reviewing components with that label. It shouldn't merely be a tag. You should also consider adding component labels such as reviewed, so someone can tell if something was reviewed. The possibilities are endless, but this may mean you need to rethink the component labels you have, or should create.

Should this component label apply to only my applications, or all applications for this organization?

> We like to refer to this as the scope of a component label. A good way to look at scope is the concept of macro and micro. At the macro level we have an organization that may have thousands of applications. If I assign this label to the component at the macro level (organization) it means the label will be seen by all applications under that organization. A component label at the root organization will have an even larger scope. That's a pretty sweeping change, and it could be the right thing to do. However, you might actually be better served by considering the impact at the micro level. In this case, maybe this is a component label that should only be applied to a particular application.

> ⓘ Only component labels that have been created at the root organization, can be assigned to the scope of root organization or organization level. Only components created at the organization can be assigned to the scope of organization.

If I assign this component label, is it part of a policy, and does it escape certain violations?

> Conditions in a policy can include values for component labels. In many cases, this is best used as a way to prevent a certain component from violating the condition. For example, I could have a policy that requires a component to be no older than three years. However, a safe, and commonly used component is four years old. If I have a process built around reviewing a case like this, where an exception would be valid, I could first place component labels to identify the component to be reviewed for an exception, and then another component label once that exception is approved (or disapproved). In this scenario, if I have built policies correctly, including allowing them to flow through certain stages, even with a violation, development can continue. This of course should occur simultaneously to a review/exception process. It's important to consider scenarios like this when creating component labels, as well as policy.

There are more questions regarding component labels that should be asked as well, but many of these you will discover as you develop your own processes. The key is that component labels can be deceivingly simple, given their implementation in other systems. Now that you have the word of warning, let's take a look at how to assign a label.

Assigning a Label

When assigning a label, you will only have access to those component labels created specifically for the application, or for that application's organization, or the root organization. Given this, if you don't see a label you need, speak with whomever is responsible for managing the component labels for your application and parent organizations.

1. Access the *Application Composition Report* for your application.
2. Click on the *Policy Violations* tab.
3. Click a component you wish to assign a label to. The *Component Information Panel (CIP)* displays.
4. Click the *Labels* option from the CIP menu. Two boxes will be displayed:

a. The *Applied* box on the left represents labels that have been assigned to the component already.
b. The *Available* box on the right displays all labels.

5. Clicking on the button on the right side of a label will move to the opposite side. Hovering over a label displays the description.
6. Click on the **+** button on the right side of a label in the *Available* list to assign the label to the component.
7. Click on the **-** button on the right side of a label in the *Applied* list to remove the label from the component.

If the label was created at the organization level, you will be presented with two options:

- Assign the label to the current component in the current application
- Assign the label to the current component within the organization, so that all applications within the organization gain access to the label assigned.

If the label was created at the root organization level, you will be presented with three options:

- Assign the label to the current component in the current application
- Assign the label to the current component within the organization, so that all applications within the organization gain access to the label assigned.
- Assign the label to the current component within the root organization, so that all applications within all organizations gain access to the label assigned.

Waivers

Overview

If you look at policy violations as a pain point preempting the flow of work, you are likely going about your evaluations and policy in the wrong way. In fact, if you saw the title of this section, hoping to find a way past policy violations, there may be a couple issues.

First, your policies should be designed to encourage workflow and communication. If development is being stopped regularly, you might want to revisit your policies, refining them so they present the possibilities for making better choices, not simply halting work altogether.

Second, and perhaps most importantly, there are not false positives when it comes to policy violations. If you are looking for ways just to get past a violation, you've circumvented the goal of policy creation. Why not simply not make it a policy? Better yet, this might be a problem with policy, or the perception of what should and should not be in your application.

So, excluding those possibilities, and working with the idea that you are here to find a way to accommodate the exceptions you may run across, waivers can help.

Use Case for Waivers

Let's say, you have a component that's violating a policy, which has been created in an organization that houses your application. It's a great policy, in fact, one of many great policies. Unfortunately, one of your applications has a component that is violating a policy.

The problem is, that this policy just doesn't accurately line up to the implementation of this particular application. Your application has a security vulnerability that can be exploited when it connects to the internet. It's a pretty severe vulnerability, but benign given that your application is internal, and doesn't even have the ability to connect to the Internet.

What should you do?

You could…

- Change the Security Status to *Not Applicable*.
- Adjust the policy to be less stringent.
- Use labels in a condition, providing an escape for exceptions.

… or, you could add a waiver.

That is, by adding a waiver, you indicate that this particular component, either in the scope of this application (what we would do in our example), or all applications for the organization, is waived from this particular policy. In fact, if you desired you could even specify that you want to waive all components (within scope of an application or organization) from a policy.

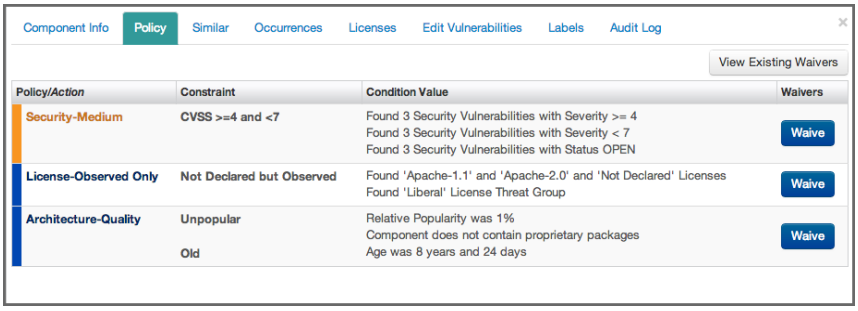
Now the important thing to take note of here, is that while this waiver seems to be the answer to policy violations strife, you are actually waiving an entire policy. This means all constraints, and in turn, all conditions. It's no surprise why that should be something that's limited. For this reason, before waiving something, it's good practice to review some alternatives. A waiver very much does allow you to simply bypass all controls.

OK, we've harped enough on the warnings. The benefits of waivers can be just as numerous. This is because even with endless customizations, you will encounter situations where a policy just doesn't apply. It's important to take a look at the full range of waiver functionality, so let's look at how to add, view, and when necessary, remove a waiver.

Adding a Waiver

1. Access an Application Composition Report.
2. Navigate to the *Policy Violations* tab on the report, and click on a component that has policy violations. This displays the Component Information Panel (CIP).

3. In the CIP, click the *Policy Violations* tab. This displays the list of Policy Violations for the Component:



4. Click the *Waive* button next to the violation you wish to waive. A dialog displays:

5. There are two options at this point, and each should be carefully considered:
    a. The first option defines the scope for the waiver. This can be either the current application or all applications for the organization.
    b. The second option defines the target of the waiver. That is the currently selected component, or all components.
6. Enter an optional *Comment*, and then click the *Waive* button to process the waiver.

When processing a waiver, depending on the options that are chosen, you can effectively waive a policy for all components, for all applications in an organization. Since this will waive the entire policy, not just this violation, it may be a good idea to ensure adjusting the policy would not provide a solution that is more visible to all users.

Viewing and Removing a Waiver

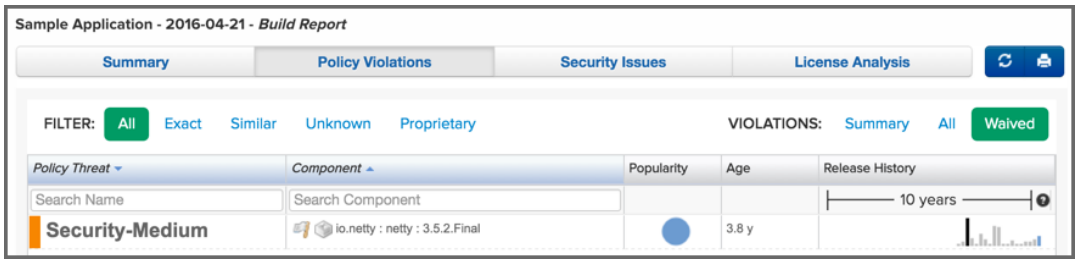
As we mentioned previously, component violations can be waived for a single component in a single application, all the way up to all components in all applications. This means, that a violation for a component

in your application could have been waived elsewhere. A good practice when reviewing the Application Composition Report is to check and see what violations have been waived for components in your application. Here are a couple examples of why this is important:

- Scenario 1: A violation for a component has been waived, and the component has additional violations. Depending on the view selected, at least one of these additional violations will be displayed.
- Scenario 2: The only violation for a component has been waived. Given that the component has no additional violations, it will be moved into the None policy threat group (light blue) in the Summary view, while the other views will only show the waived violation.

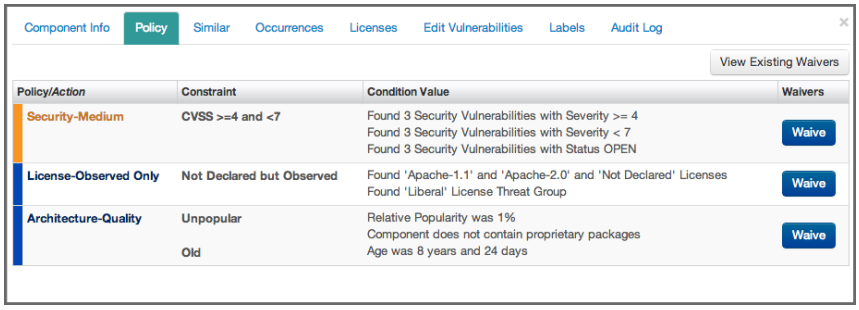
To view waived violations for your components:.

1. Access an *Application Composition Report*.
2. Navigate to the *Policy Violations* tab on the report. Just above the list of components, and to the right of the report, you will see three options in the *Violations* filter:



   a. *Summary* - this is the default view of the Policy tab. It is important to note that even though this view will display all components, only the highest threat violation per component is displayed. In this view, components with waived violations may have been moved to the *None* policy threat group (light blue).
   b. *All* - clicking this filter option will display every violation for all components in your application. This may result in the appearance of duplicates in the component list. Violations that have been waived will be indicated by a white flag icon.
   c. *Waived* - clicking this filter option will display only the waived violations. In this view, you will only see those components where violations have been waived. Each component will have a white flag icon, and it is likely you will not see all components. This view may also produce the appearance of duplicated components.

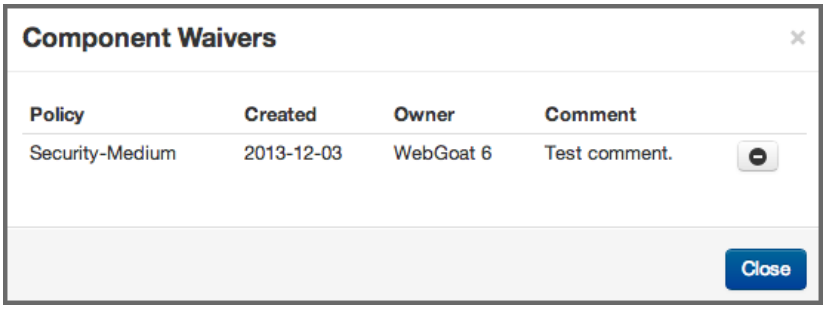
3. Click on a component to display the Component Information Panel (CIP):



4. At the top of the of the component list, click on the *View Existing Waivers* button. A modal displays showing all the waivers for the component, as well as the associated descriptions.

   To remove a waiver:

5. Click the remove icon (−):



6. A message asks you to confirm this removal. Click the *Remove* button to continue.

> Because some waivers can be set for all applications, and even all components, it's important to understand the impact of removing a waiver. Be sure to verify with the application or organization owner, the intended scope of the waiver.

Policy Reevaluation

You will likely find a number of consistent themes through this documentation. One of these is that regular policy review and refinement should be part of your company's approach to policy management.

Accomplishing this successfully could potentially mean regularly rebuilding applications or publishing them to repositories several times over. Not to mention that in the case of waiting for builds, you might wait hours before an evaluation is able to run.

While there are a variety of reasons this can happen (e.g. build times are slow), the important thing is that access to the new results could be delayed. If you've made a change to policy you won't be able to tell if that made a difference. Then it's highly likely, you'll need to make another change, and then wait again. Luckily there is an alternative which allows you to reevaluate the results of an evaluation.

Using the existing component information from the most recent evaluation against the current policies - which you might have changed since the last build and analysis - you can update an Application Composition Report.

To do this, you can use policy reevaluation to see how your changes affect the current policy. The policy reevaluation button, located in the top right of the Application Composition Report (to the left of the PDF Export/Printer icon). Simply click this button and any policy changes you've made will be considered against the data of the current report.

Of course, it's possible other data in the application could have changed, and that might not be realized until the next build. However, this will give you a good idea of how immediate policy changes impact any violations you currently have.

> Policy Reevaluation will not enact any actions you may have attached to your policies.

PDF Report

Not everyone will have access to the IQ Server or any of the integrated enforcement points, and in turn, any of the associated reports. However, certain individuals or teams would likely still benefit from the information the Application Composition Report provides. Even if that's not your particular situation, you may reach a point where you would like to produce an archive of a Application Composition Report for historical and audit purposes. Given this need, every report you produce can be converted into a PDF.

Though the information presented in both the web application and the PDF are nearly identical, there are a few difference, mainly formed out of the contrasting visual and layout capabilities of a web application versus PDF. Below, we'll discuss how to create this PDF version as well as highlight some of the differences between the two.

Creating the PDF

With the Application Composition Report open, find the set of blue icons in the top right. While the first icon is related to reevaluating the report, the button on the right allows you to create a PDF version of the report. Simply, click on this button  and your browser will prompt you to download a PDF version of the report.

> The report filename will be unique each time you use the button. However, in general the report filename will use the following pattern: **`applicationName-stageName-timestamp.pdf`**.

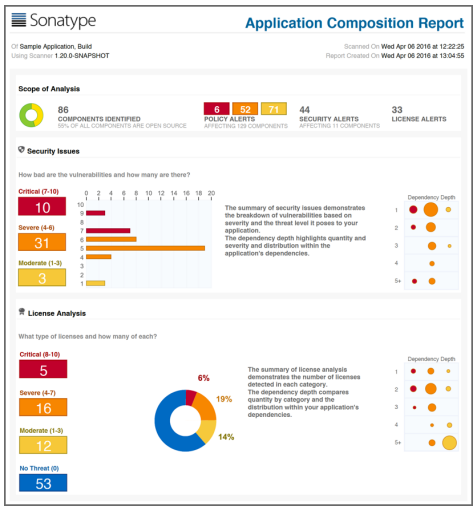
Reviewing the PDF

The information provided by the PDF is identical to the information that is provided within the Application Composition Report in the application user interface. This includes the *Summary*, *Policy*, *Security Issues*, and *License Analysis* tabs. Within the PDF, the order of information is presented top to bottom, following the logic

of the report tabs from left to right. With the exception of the first page, which provides the *Summary*, each section has a label to indicate the corresponding tab of the Application Composition Report:

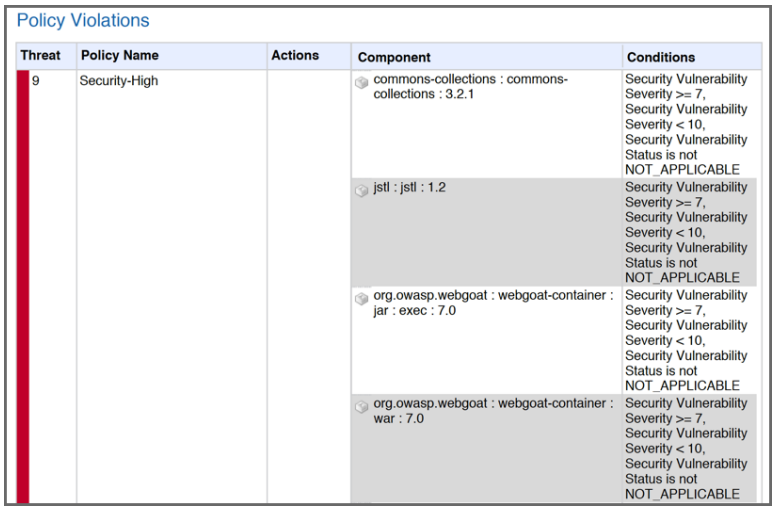
Summary

The summary section is identical to the HTML version of the report:



Policy Violations

The Policy Violations section displays the details for all scanned components. This matches the data displayed in the Policy tab of the Component Information Panel (CIP). It should be noted, that depending on the number of violations in your application, this section could be very long:

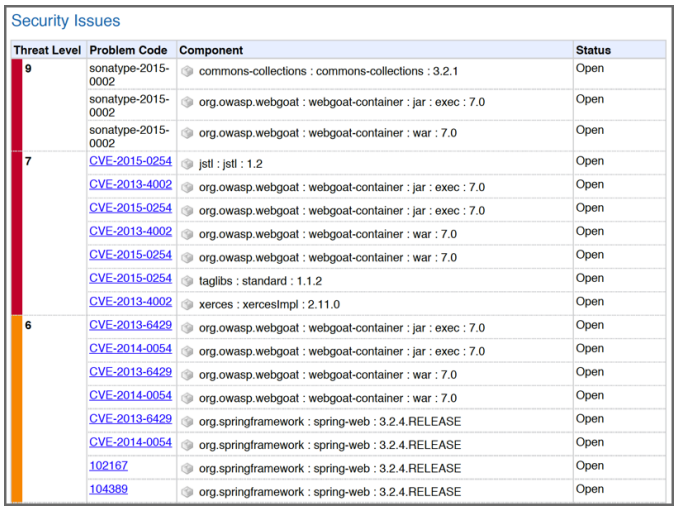


Policy Violations

Threat	Policy Name	Actions	Component	Conditions
9	Security-High		commons-collections : commons-collections : 3.2.1	Security Vulnerability Severity >= 7, Security Vulnerability Severity < 10, Security Vulnerability Status is not NOT_APPLICABLE
			jstl : jstl : 1.2	Security Vulnerability Severity >= 7, Security Vulnerability Severity < 10, Security Vulnerability Status is not NOT_APPLICABLE
			org.owasp.webgoat : webgoat-container : jar : exec : 7.0	Security Vulnerability Severity >= 7, Security Vulnerability Severity < 10, Security Vulnerability Status is not NOT_APPLICABLE
			org.owasp.webgoat : webgoat-container : war : 7.0	Security Vulnerability Severity >= 7, Security Vulnerability Severity < 10, Security Vulnerability Status is not NOT_APPLICABLE

Security Issues

The Security Issues section displays a breakdown of all security issues found in the scan of the application, matching what is displayed in the HTML version of the report:

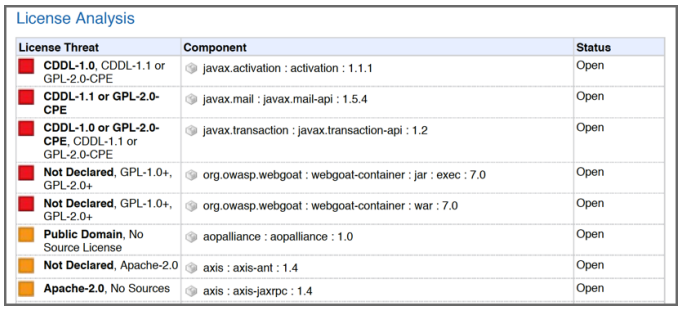


Security Issues

Threat Level	Problem Code	Component	Status
9	sonatype-2015-0002	commons-collections : commons-collections : 3.2.1	Open
	sonatype-2015-0002	org.owasp.webgoat : webgoat-container : jar : exec : 7.0	Open
	sonatype-2015-0002	org.owasp.webgoat : webgoat-container : war : 7.0	Open
7	CVE-2015-0254	jstl : jstl : 1.2	Open
	CVE-2013-4002	org.owasp.webgoat : webgoat-container : jar : exec : 7.0	Open
	CVE-2015-0254	org.owasp.webgoat : webgoat-container : jar : exec : 7.0	Open
	CVE-2013-4002	org.owasp.webgoat : webgoat-container : war : 7.0	Open
	CVE-2015-0254	org.owasp.webgoat : webgoat-container : war : 7.0	Open
	CVE-2015-0254	taglibs : standard : 1.1.2	Open
	CVE-2013-4002	xerces : xercesImpl : 2.11.0	Open
6	CVE-2013-6429	org.owasp.webgoat : webgoat-container : jar : exec : 7.0	Open
	CVE-2014-0054	org.owasp.webgoat : webgoat-container : jar : exec : 7.0	Open
	CVE-2013-6429	org.owasp.webgoat : webgoat-container : war : 7.0	Open
	CVE-2014-0054	org.owasp.webgoat : webgoat-container : war : 7.0	Open
	CVE-2013-6429	org.springframework : spring-web : 3.2.4.RELEASE	Open
	CVE-2014-0054	org.springframework : spring-web : 3.2.4.RELEASE	Open
	102167	org.springframework : spring-web : 3.2.4.RELEASE	Open
	104389	org.springframework : spring-web : 3.2.4.RELEASE	Open

License Analysis

The License Analysis section displays a breakdown of all license issues found in the scan of the application, matching what is displayed in the HTML version of the report. It should be noted that depending on your license threat groups, and license assignments, this section of the report could be very long:

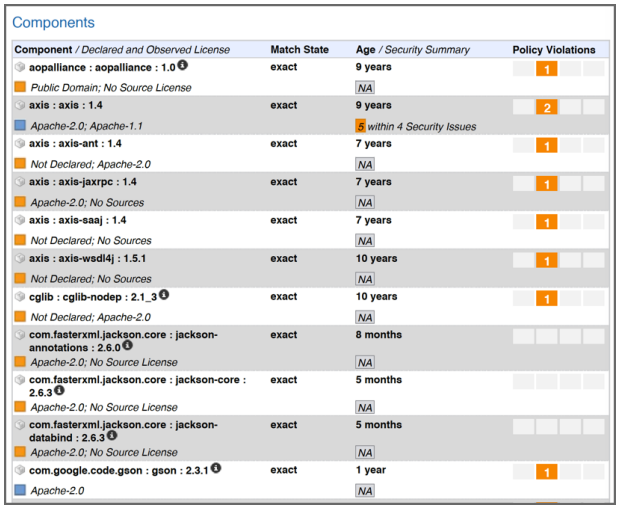


License Analysis

License Threat	Component	Status
CDDL-1.0, CDDL-1.1 or GPL-2.0-CPE	javax.activation : activation : 1.1.1	Open
CDDL-1.1 or GPL-2.0-CPE	javax.mail : javax.mail-api : 1.5.4	Open
CDDL-1.0 or GPL-2.0-CPE, CDDL-1.1 or GPL-2.0-CPE	javax.transaction : javax.transaction-api : 1.2	Open
Not Declared, GPL-1.0+, GPL-2.0+	org.owasp.webgoat : webgoat-container : jar : exec : 7.0	Open
Not Declared, GPL-1.0+, GPL-2.0+	org.owasp.webgoat : webgoat-container : war : 7.0	Open
Public Domain, No Source License	aopalliance : aopalliance : 1.0	Open
Not Declared, Apache-2.0	axis : axis-ant : 1.4	Open
Apache-2.0, No Sources	axis : axis-jaxrpc : 1.4	Open

Components

This section brings together information from all the others. It displays the highest security issue identified (and the associated CVSS score), any declared and/or observed licenses (and the highest threat level of the associated), the match state, age, and the policy violation counts for each threat level band (red, orange, yellow, and blue) for each component. In most cases this section can be used as a detailed bill of materials:



Components

Component / Declared and Observed License	Match State	Age / Security Summary	Policy Violations
aopalliance : aopalliance : 1.0	exact	9 years	1
Public Domain; No Source License		NA	
axis : axis : 1.4	exact	9 years	2
Apache-2.0; Apache-1.1		5 within 4 Security Issues	
axis : axis-ant : 1.4	exact	7 years	1
Not Declared; Apache-2.0		NA	
axis : axis-jaxrpc : 1.4	exact	7 years	1
Apache-2.0; No Sources		NA	
axis : axis-saaj : 1.4	exact	7 years	1
Not Declared; No Sources		NA	
axis : axis-wsdl4j : 1.5.1	exact	10 years	1
Not Declared; No Sources		NA	
cglib : cglib-nodep : 2.1_3	exact	10 years	1
Not Declared; Apache-2.0		NA	
com.fasterxml.jackson.core : jackson-annotations : 2.6.0	exact	8 months	
Apache-2.0; No Source License		NA	
com.fasterxml.jackson.core : jackson-core : 2.6.3	exact	5 months	
Apache-2.0; No Source License		NA	
com.fasterxml.jackson.core : jackson-databind : 2.6.3	exact	5 months	
Apache-2.0; No Source License		NA	
com.google.code.gson : gson : 2.3.1	exact	1 year	1
Apache-2.0		NA	

In some cases a URL for the project is provided. This is indicated by an information icon .

Impact of Improved JavaScript Reporting

As of July 2018 we will be enacting improvements to our JavaScript reporting, the impacts of which are discussed here.

Existing Component Policy Waivers/Labels and Security Vulnerability Overrides

Existing configurations of:

- Component-specific Policy Waivers (see page 219)
- Component Labels (see page 168)
- Security Vulnerability Overrides (see page 207)

to individual JavaScript files will not be carried over to aggregated JavaScript components and instead will need to be reapplied. This will hold even for an aggregated JavaScript component that only consists of one JavaScript file.

This improvement will allow remediation efforts to be applied to entire JavaScript components associated with multiple JavaScript files rather than just one JavaScript file at a time.

Reduced Overlapping Matches

The same JavaScript file can be matched to several different ecosystems. For example, a JavaScript file from the package jquery version 3.0.0 can be associated to multiple components with different formats. In addition to aggregating JavaScript files, overlapping matches will also be removed.

Policy Violations, Security Issues, and License Threats

Due to these changes you may observe a different number of Policy Violations, Security Issues, and License Threats for the following reasons:

1. Since existing Component-specific Policy Waivers and Security Vulnerability Overrides will need to be reapplied, until this is done it may increase the number of Policy Violations and Security Vulnerabilities (see page 62).
2. Since existing Component Labels will need to be reapplied (see page 217), which may be tied to specific Policies (see page 146), this may also alter the number of Policy Violations.
3. The aggregation of JavaScript files and removal of overlapping matches may decrease the overall number of JavaScript components (the noise) resulting in a decreased number of Policy Violations, Security Issues, and License Threats.

Additionally, if there are differences, then the corresponding Summary statistics will also differ.

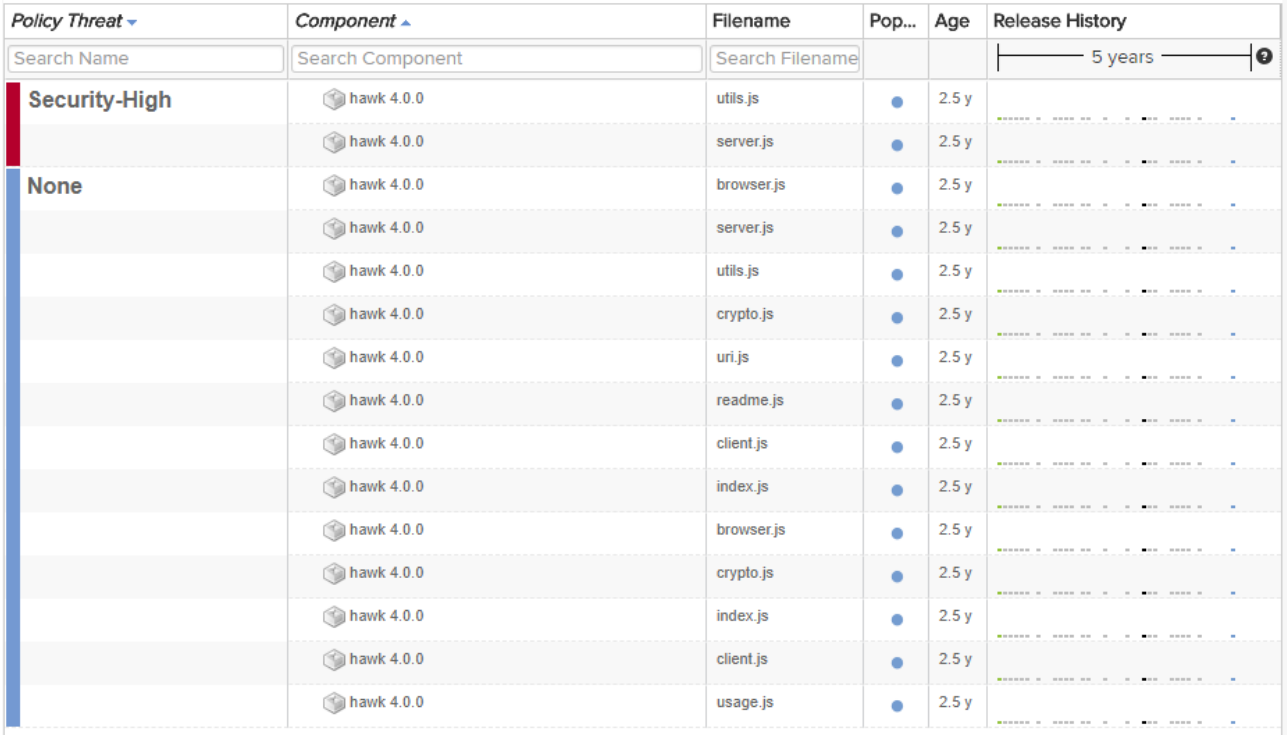
One consequence to keep in mind of the potentially different Policy Violations is that you may see an increased number of unsuccessful builds. Due to this you may want to disable the Fail action on Policies for the Build stage (see page 153) until you have reapplied your component Policy Waivers/Labels and Security Vulnerability Overrides. Additionally, you may also want to enable Notifications (see page 156) on your Policies to be able to more easily spot Policy Violations that have appeared due to these changes.

Example Application Report Before and After

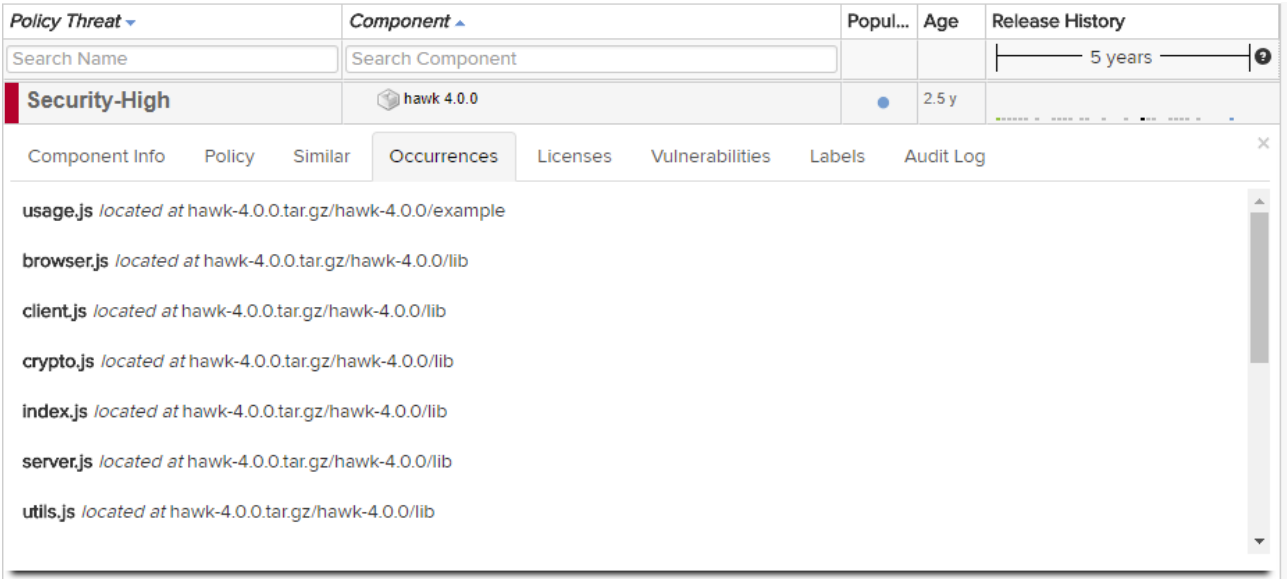
For this example we look at the JavaScript package hawk version 4.0.0.

Policy Violations

Before these changes, each unique identified JavaScript file is represented by one row i.e.



After these changes, one row will be able to represent a multitude of these files i.e.



Here all of the JavaScript files have been aggregated to the same JavaScript component and are listed on the Occurrences (see page 203) tab within the Component Information Panel (see page 202).

Security Issues

An aggregate JavaScript component will retain all unique Security Vulnerabilities belonging to its aggregated JavaScript files.

In this example, two JavaScript files share the same Security Vulnerability, so before these changes we would see

Threat Level	Problem Code	Component	Filename	Status
Search Level	Search Code	Search Component	Search Filename	Search Status
7	SONATYPE-2016-0009	hawk 4.0.0	utils.js	Open
	SONATYPE-2016-0009	hawk 4.0.0	server.js	Open

and after these changes they are aggregated to the same JavaScript component, and so that Security Vulnerability is only displayed once for it i.e.

Threat Level	Problem Code	Component	Status
Search Level	Search Code	Search Component	Search Status
7	SONATYPE-2016-0009	hawk 4.0.0	Open

License Analysis

In this example, the number of unique components remains the same, and so the number of rows on the License Analysis (see page 209) tab is unaltered.

License Threat	Component	Status
Search Licenses	Search Component	Search Status
BSD-3-Clause, Not Supported	hawk 4.0.0	Open

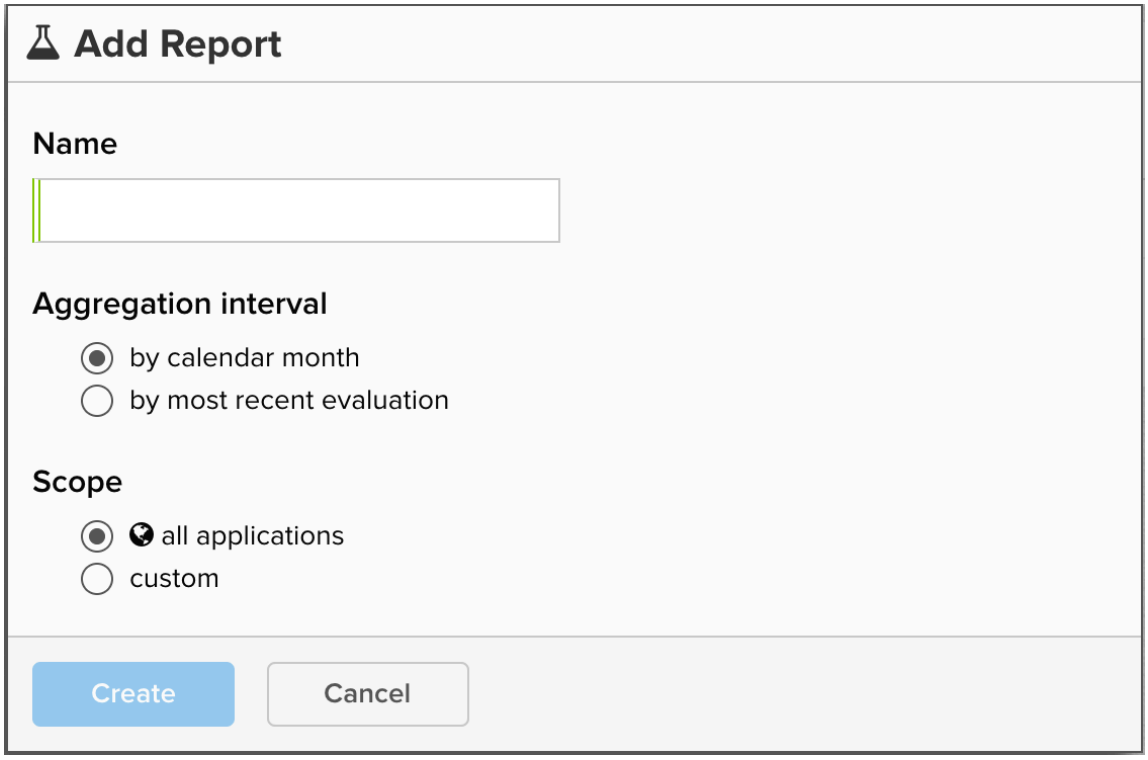
Filename

A JavaScript component may be associated with multiple JavaScript files after these changes and so the Filename column is now obsolete and omitted from the Application Report.

Success Metrics

Starting with the 1.33 release, IQ Server provides access to Success Metrics. These are reports that demonstrate the value of IQ Server, highlighting metrics like the average number of policy violations per application. They can also show the progress your organization is making by presenting changes in metrics over time.

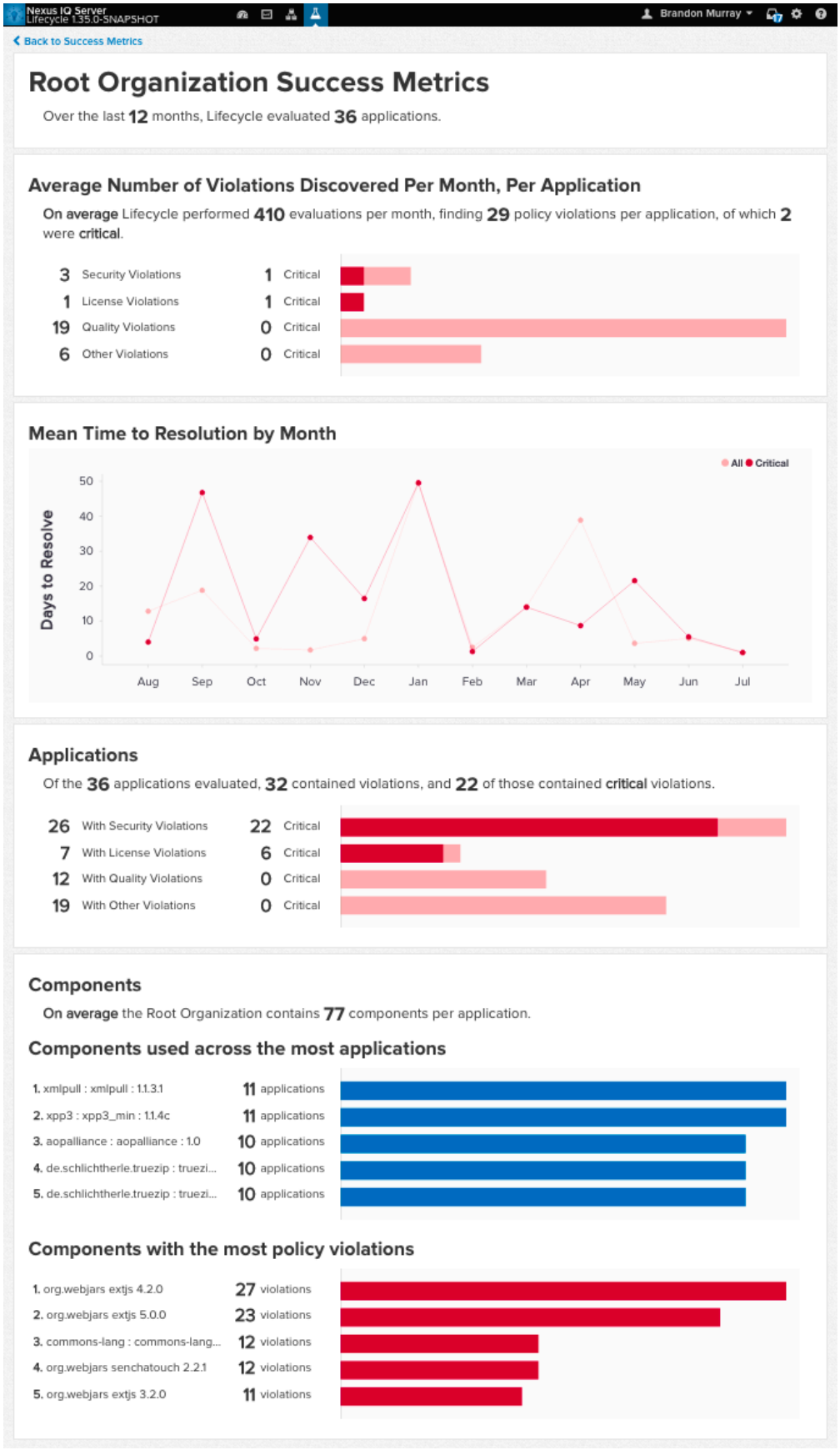
To access the reports, click the Success Metrics icon in the toolbar. On the *Success Metrics* page, you have the option to add a new Success Metrics Report as well as a list of all Success Metrics Reports which have previously been added. Upon first usage, click the *Add Report* button near the top-right corner to open the *Add Report* modal window.



Here you can specify the aggregation interval - including evaluations up to the end of the previous month, or including the most recent evaluations as well. You can also specify which applications and organizations to include in each report. By default, all applications to which you have access are included. Alternatively, you can specify a custom set.

> Success Metrics Reports are defined on a per-user basis. A report created by one user will not be visible to others.

Once a new report is configured, it will appear on the *Success Metrics* page where it can be selected to navigate to a *Success Metrics Report* page generated from the associated set of applications. That page is shown below:


Nexus IQ Server
Lifecycle 1.35.0-SNAPSHOT
Brandon Murray
Back to Success Metrics
Root Organization Success Metrics
Over the last 12 months, Lifecycle evaluated 36 applications.
Average Number of Violations Discovered Per Month, Per Application
On average Lifecycle performed 410 evaluations per month, finding 29 policy violations per application, of which 2 were critical.
3 Security Violations 1 Critical
1 License Violations 1 Critical
19 Quality Violations 0 Critical
6 Other Violations 0 Critical
Mean Time to Resolution by Month
All Critical
Days to Resolve
50
40
30
20
10
0
Aug Sep Oct Nov Dec Jan Feb Mar Apr May Jun Jul
Applications
Of the 36 applications evaluated, 32 contained violations, and 22 of those contained critical violations.
26 With Security Violations 22 Critical
7 With License Violations 6 Critical
12 With Quality Violations 0 Critical
19 With Other Violations 0 Critical
Components
On average the Root Organization contains 77 components per application.
Components used across the most applications
1. xmlpull : xmlpull : 1.1.3.1 11 applications
2. xpp3 : xpp3_min : 1.1.4c 11 applications
3. aopalliance : aopalliance : 1.0 10 applications
4. de.schlichtherle.truezip : truezi... 10 applications
5. de.schlichtherle.truezip : truezi... 10 applications
Components with the most policy violations
1. org.webjars extjs 4.2.0 27 violations
2. org.webjars extjs 5.0.0 23 violations
3. commons-lang : commons-lang... 12 violations
4. org.webjars senchatouch 2.2.1 12 violations
5. org.webjars extjs 3.2.0 11 violations

⚠ Larger data sets may take considerable time to load the first time you access a report.

The first section is the *Summary* tile that shows the average number of evaluations done per month and the average number of violations per application, including how many of those violations are critical. That information is broken down even further by policy type - security, license, quality, or other.

The second section is the *Mean Time to Resolution* (*MTTR*) *by Month* tile. It shows the average age of resolved violations for the last year. This information is further broken down by their threat level.

The third section is the *Applications* tile that breaks down how many of the applications included in the report have violations and of those how many are critical.

The last section is the *Components* tile that breaks down which components are used across the most applications and which components have the most policy violations.

If needed, Success Metrics can be turned off by a System Administrator via the System Preferences menu in the toolbar.

Success Metrics in Action:

IQ Server Integrations

There are several integrations for IQ Server available from the Sonatype team. The table below provides a brief introduction of our plugins and provides links to additional information in our Nexus Integrations[107] space.

Integration	Description
IDEA[108]	IQ for IDEA provides component analysis for both the Community and Ultimate edition of IntelliJ IDEA.
Eclipse[109]	The Sonatype CLM for Eclipse plugin lets you perform component analysis, inspect component details, and fix issues all from your IDE.
Visual Studio[110]	IQ for Visual Studio provides component analysis for both the Community, Professional and Enterprise versions of Visual Studio.
Nexus Repository Manager 2.x[111]	IQ for Nexus Repository Manager 2.x allows you to integrate IQ Server's policy management and component intelligence features with proxy repositories in Nexus Repository Manager Pro.
Nexus Repository Manager 3.x[112]	IQ for Nexus Repository Manager 3.x allows you to integrate IQ Server's policy management and component intelligence features with proxy repositories in Nexus Repository Manager Pro.
Bamboo[113]	IQ for Bamboo analyzes the components used in your software development for security and license characteristics.
Hudson/ Jenkins 1.x[114]	IQ for Hudson/Jenkins 1.x evaluates the project workspace after a build for all supported component types, creates a summary file about all the components found and submits that to the IQ Server.
Jenkins 2.x[115]	The Nexus IQ for Jenkins 2.x plugin provides full component intelligence and the ability to run policy against your application.
CLI[116]	While tools offering full integration for evaluations are provided (e.g. Nexus Repository Manager, Eclipse, and Hudson/Jenkins), any application can be evaluated against your policies using the Nexus IQ CLI.

107 https://help.sonatype.com/display/NXI/Nexus+Integrations
108 https://help.sonatype.com/display/NXI/IQ+for+IDEA
109 https://help.sonatype.com/display/NXI/Sonatype+CLM+for+Eclipse
110 https://help.sonatype.com/display/NXIQ/2017/06/22/IQ+for+Visual+Studio+Extension
111 https://help.sonatype.com/display/NXI/IQ+Server+and+NXRM+2.x
112 https://help.sonatype.com/display/NXI/IQ+Server+and+NXRM+3.x
113 https://help.sonatype.com/display/NXI/Nexus+IQ+for+Bamboo
114 https://help.sonatype.com/pages/viewpage.action?pageId=329725
115 https://help.sonatype.com/display/NXI/Nexus+IQ+for+Jenkins
116 https://help.sonatype.com/display/NXI/Nexus+IQ+CLI

Maven[117]	Sonatype CLM for Maven lets you evaluate any Maven-based software projects in the same way as our integrated tools providing access the same robust reporting features no matter what toolset you use. It can be run on a command line interface and executed on any continuous integration server, as well as a number of popular IDEs.
SonarQube[118]	Sonatype CLM for SonarQube uses code analysis to provide better component usage.

Go to Integration Requirements[119] to see what's needed for each integration. See Download and Compatibility (see page 53) to get the latest version of IQ Server integrations.

REST APIs

Using REST API calls, the IQ Server provides functionality to create and update applications, as well as retrieve values for policy violations.

These APIs have been designed for system-to-system functionality; however, examples are provided using the HTTP client cURL. Following along, you can initiate the described API REST request via the command line tool.

REST APIs are versioned. This document represents the most recent version. If you plan to use any of these, we highly recommend updating to the latest version of the IQ Server to ensure compatibility.

Available APIs:

117 https://help.sonatype.com/display/NXI/Sonatype+CLM+for+Maven
118 https://help.sonatype.com/display/NXI/Sonatype+CLM+for+SonarQube
119 https://help.sonatype.com/display/NXI/Integration+Requirements

Component Search REST APIs - v2

The Component Search API allows you to find a particular component, as well as get information back about that component. Using GET requests it allows you to retrieve component information such as application ID, application name, report URL, component hash, component coordinates, and the highest threat level of the policy violations (for the found component).

Below, we've provided an example of the GET request. We've done this using the HTTP client cURL. Of course, you could use any HTTP client tool. Additionally, to help demonstrate use of the API, we've broken out the various pieces for this request and provided an example of data that is retrieved.

> Compared to the other APIs, Component Search is fairly simple. However, you should have some basic information about the component coordinates, as well as the stage (e.g. Build) where the component was analyzed.

Searching for Components

First, make sure your IQ Server is started. Also, as we mentioned, you will need to have evaluated at least one application. With those two things completed, let's take a look at the GET API.

```
GET /api/v2/search/component
```

Now, in addition to this, you will need to add two parameters - the **stage**, and then add your **search parameters**.

Stage

Typically the stage represents the development lifecycle of your product. There are four stages that are currently supported.

These include:

- **`build`**
- **`stage-release`**
- **`release`**
- **`operate`**

Entering any of these for the stage ID will pull from that specific stage's evaluation data.

Next up, you need to set the component search parameters using any combination of these options:

Search Parameters

The search parameters can either represent a direct hash of a component or a **componentIdentifier** object.

- **hash** - the component hash (e.g. hash=1234567890).
- **componentIdentifier** - this is an object that contains the the coordinates of the component and its format.
- **format** - this is the format of the component (e.g. "maven" or "a-name" or "python").
- **coordinates** - this object contains the name/value pairs for identifying coordinates (e.g. artifactId, groupId, version, extension, and classifier).

If you are working with a command line utility like curl, remember to URL encode the componentIdentifier payload.

Maven example

Now, let's look at an example. Consider a case where we wanted to find all components within the group ID "tomcat", for any applications evaluated during the Build stage. Using the information above, as well as cURL and an encoded URL, here is what we would have…

```
curl -u admin:admin123 -X GET "http://localhost:8070/api/v2/search/component?
stageId=build&componentIdentifier=%7B%22format%22%3A%22maven%22%2C%22coordinates%22%3A%7B%22groupId%22%3A%2
2tomcat%22%2C%22artifactId%22%3A%22*%22%2C%22version%22%3A%22*%22%2C%22extension%22%3A%22*%22%2C%22classifi
er%22%3A%22*%22%7D%7D"
```

Of course the above is an encoded URL, so just for a bit of help, here is what a non-encoded URL would look like. This should help you identify the JSON in the example above.

```
"http://localhost:8070/api/v2/search/component?
stageId=build&componentIdentifier={"format":"maven","coordinates":
{"groupId":"tomcat","artifactId":"*","version":"*","extension":"*","classifier":"*"}}"
```

JavaScript example

We call the coordinate system for JavaScript "*a-name*", which is short for authoritative name. If we want to search for any applications that contain the JavaScript package vizion, version 0.1.0 we would use the following componentIdentifier object

```
"http://localhost:8070/api/v2/search/component?stageId=build&componentIdentifier={"format":"a-
name","coordinates":{"name":"vizion","qualifier":"","version":"0.1.0"}}"
```

Python example

If we want to search for a python pypi package pyOpenSSL in the version 17.0.0 we would use the below componentIdentifier

```
"http://localhost:8070/api/v2/search/component?
stageId=build&componentIdentifier={"format":"pypi","coordinates":{"name":"pyOpenSSL","qualifier":"py2.py3-
none-any","version":"17.0.0","extension":"whl"}}"
```

Included in our cURL example is the default IQ Server location (localhost:8070) as well as the default username and password for the admin account. Your account credentials may vary, but are necessary in order for the request to be processed. The results provided will be filtered based on the permissions of the credentials you use. For example, if you are not included in at least the developer role for an application, no results will be returned for that application. Given this, some results may be obscured.

Output

The response from the API is a JSON object. Assuming we ran the maven query searching for tomcat components, the JSON response will list all applications containing the tomcat component. An example output:

```
{
        {
   "criteria":{
      "stageId":"build",
      "hash":null,
      "componentIdentifier":{
         "format":"maven",
         "coordinates":{
            "groupId":"tomcat"
                        "artifactId":"*",
                        "version":"*",
                        "extension":"*",
                        "classifier":"*"
         }
      }
   },
   "results":[
```

```
{
   "applicationId":"MyApp-1234",
   "applicationName":"My Application 2",
   "reportUrl":"http://localhost:8070/ui/links/appl
   ication/MyApp-1234/report/c81991938f304f30bc139e
   a13cf93cd5",
   "hash":"1249e25aebb15358bedd",
   "componentIdentifier":{
      "format":"maven",
      "coordinates":{
         "artifactId":"tomcat-util",
         "classifier":"",
         "extension":"jar",
         "groupId":"tomcat",
         "version":"5.5.23"
      }
   }
         }
```


The data above was formatted to make it a bit more readable.

Using Wildcards

Of course, you may come across instances where you want to produce more results with less specific component details. If this is the case, the Component Search API does support the use of wildcards when searching using the GAVEC (coordinates).

If you are familiar with the coordinates policy condition, it follows the exact same logic. You can read more on the Coordinates condition in Policy Management (see page 148).

Component Details API - v2

The Component Details API provides all available (to Sonatype) security vulnerability, license data, age, and popularity information for a specified component. What is not included, is any information related to policy violations for an evaluated application.

If you are looking for component information for a component that has been evaluated as part of an application, please see the Component Details by Report API.

This API uses **POST** REST resource

Below, we have provided a step-by-step example using the HTTP client cURL, though any HTTP client could be used.

Step 1: Get the Component HASH or Component Identifier

Depending on the type of component, and the information you have, the API allows you to specify the component hash, the component identifier, or both. In our example we'll be searching using Maven coordinates.

> If desired you can specify more than one component.

Step 2 - Submit the Specified Component to Retrieve Details

First let's take a look at the POST resource:

```
POST api/v2/components/details
```

You will also need to include JSON data specifying the component information you are providing.

```
{
  "components": [
    {
      "hash": null,
      "componentIdentifier": {
        "format": "maven",
        "coordinates": {
          "artifactId": "tomcat-util",
          "extension": "jar",
          "groupId": "tomcat",
          "version": "5.5.23"
         }
      }
    }
  ]
}
```

Putting this together with the cURL command, as well as including the IQ Server URL for the POST resource path, you should have something that looks like this:

```
curl -u admin:admin123 -X POST -H "Content-Type: application/json" -d '{"components":[{"hash":
null,"componentIdentifier": {"format":"maven","coordinates": {"artifactId":"tomcat-
```

```
util","extension":"jar","groupId":"tomcat","version":"5.5.23"}}}]}' 'http://localhost:8070/api/v2/
components/details'
```

The IQ Server will then respond with the component details. An example is provided below.


```
{
   "componentDetails":[
      {
         "component":{
            "hash":"1249e25aebb15358bedd",
            "componentIdentifier":{
               "format":"maven",
               "coordinates":{
                  "artifactId":"tomcat-util",
                  "classifier":"",
                  "extension":"jar",
                  "groupId":"tomcat",
                  "version":"5.5.23"
               }
            }
         },
         "matchState":"exact",
         "catalogDate":"2008-01-29T01:45:22.000-05:00",
         "relativePopularity":100,
         "licenseData":{
            "declaredLicenses":[
               {
                  "licenseId":"Apache-2.0",
                  "licenseName":"Apache-2.0"
               }
            ],
            "observedLicenses":[
               {
                  "licenseId":"No-Sources",
                  "licenseName":"No Sources"
               }
            ]
         },
         "securityData":{
            "securityIssues":[
               {
                  "source":"cve",
                  "reference":"CVE-2007-3385",
                  "severity":4.3,
                  "url":"http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2007-3385",
                  "threatCategory":"severe"
               },
               {
                  "source":"cve",
                  "reference":"CVE-2007-5333",
                  "severity":5.0,
                  "url":"http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2007-5333",
                  "threatCategory":"severe"
```

```
            },
            {
               "source":"cve",
               "reference":"CVE-2011-2526",
               "severity":4.4,
               "url":"http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2011-2526",
               "threatCategory":"severe"
            },
            {
               "source":"cve",
               "reference":"CVE-2012-0022",
               "severity":5.0,
               "url":"http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2012-0022",
               "threatCategory":"severe"
            },
            {
               "source":"osvdb",
               "reference":"37071",
               "severity":4.3,
               "url":"http://osvdb.org/37071",
               "threatCategory":"severe"
            },
            {
               "source":"osvdb",
               "reference":"41435",
               "severity":5.0,
               "url":"http://osvdb.org/41435",
               "threatCategory":"severe"
            },
            {
               "source":"osvdb",
               "reference":"73797",
               "severity":4.4,
               "url":"http://osvdb.org/73797",
               "threatCategory":"severe"
            },
            {
               "source":"osvdb",
               "reference":"73798",
               "severity":4.4,
               "url":"http://osvdb.org/73798",
               "threatCategory":"severe"
            },
            {
               "source":"osvdb",
               "reference":"78573",
               "severity":5.0,
               "url":"http://osvdb.org/78573",
               "threatCategory":"severe"
            }
         ]
      }
   }
]
```

```
}
```

Component Evaluation REST APIs - v2

The Component Evaluation REST API allows for a single component, or multiple components, to be evaluated against a specific application and the associated policies.

This API uses the following REST resources:

- POST - to submit a component or list of components, as well as the application containing the polices the component(s) will be evaluated against.
- GET - to check the status of the evaluation and retrieve the results when completed.

For the API we provide a step-by-step example using the HTTP client cURL, though any HTTP client tool could be used. In addition we'll reference other APIs such as the one required to obtain an application's internal ID.

Step 1 - Get Component Information

First, you need to decide which components you want to evaluate. You'll need a bit of component information in the form of the GAVE (Group, Artifact, Version, and Extension) or the hash for each component you wish to evaluate.

There are a variety of ways to get the GAVE, the most common is to get the information from the repository that houses the component. For example, The Central Repository[120] has the GAVE for all components stored there.

If you wish to use the hash, Sonatype uses `sha1` for all components it processes.

> Only components known to Sonatype will provide security, license, and popularity data related to policies. However, depending on the policies for the application you choose to evaluate the component against, other policy conditions, such as match state, will help in determining if the component is known to your instance of the IQ Server.

Step 2 - Get the Application ID

Next, you will need pick the application you want to evaluate your component against. This begins with obtaining the application's public ID, which is used to retrieve the internal application ID.

120 http://search.maven.org/

The public ID can be found via the IQ Server GUI by navigating to the respective application and copying the application ID, which is located just below the application name.

This is done using the following GET REST resource from our application API:

```
GET /api/v2/applications?publicId={YourPublicId}
```

Using the cURL command, it would look like this:

```
curl -u admin:admin123 -X GET 'http://localhost:8070/api/v2/applications?publicId=MyApplicationID'
```

If you like, you can also return multiple applications by appending additional &publicId={SomeOtherPublicId} to the command above. Alternatively, if desired, all applications can always be returned by omitting the reference to the public id. This will give the public ID and internal application ID for all applications.

information will be returned (unique to your application). This has been formatted for readability:

```
{
    "applications": [
        {
            "id": "4537e6fe68c24dd5ac83efd97d4fc2f4",
            "publicId": "MyApplicationID",
            "name": "MyApplication",
            "organizationId": "bb41817bd3e2403a8a52fe8bcd8fe25a",
            "contactUserName": "NewAppContact",
            "applicationTags": [
                {
                    "id": "9beee80c6fc148dfa51e8b0359ee4d4e",
                    "tagId": "cfea8fa79df64283bd64e5b6b624ba48",
                    "applicationId": "4bb67dcfc86344e3a483832f8c496419"
                }
            ]
        }
    ]
}
```

From the information returned above, make note of the id. This is the internal application ID.

Step 3 Submit Component for Evaluation

Alright, by now you should have either the GAVE or hash for your component, as well as the internal application ID. We'll put this information together using the POST REST resource:

```
POST /api/v2/evaluation/applications/{applicationInternalId}
```

Added to this will be a JSON formatted body:

```
{
   "hash":null,
   "componentIdentifier":{
      "format":"maven",
      "coordinates":{
         "artifactId":"commons-fileupload",
         "groupId":"commons-fileupload",
         "version":"1.2.2",
         "extension":"jar"
      }
   }
}
```

If desired, multiple components can be included, but remember, the extension is required when entering components by GAVE.

Together, this should look like this:

```
curl -u admin:admin123 -X POST -H "Content-Type: application/json" -d '{"components": [{"hash":
null,"componentIdentifier": {"format": "maven","coordinates": {"artifactId": "commons-
fileupload","groupId": "commons-fileupload","version": "1.2.2","extension":"jar"}}}]}' 'http://localhost:
8070/api/v2/evaluation/applications/529b7f71bb714eca8955e5d66687ae2c'
```

A successful POST will result in JSON formatted data providing a confirmation that the evaluation was submitted.

```
{
   "resultId":"917e0c5e92a646a5b8879a40890078bc",
   "submittedDate":"2015-02-02T10:43:22.975-05:00",
   "applicationId":"529b7f71bb714eca8955e5d66687ae2c",
   "resultsUrl":"api/v2/evaluation/applications/
   529b7f71bb714eca8955e5d66687ae2c/results/
   917e0c5e92a646a5b8879a40890078bc"
}
```

Be sure to make note of resultID and "applicationID". These will be used to to check and retrieve results. If you are following along with cURL, you can simply copy the resultsURL.

Step 4 Get Evaluation Results

Now, depending on how many components you evaluated, the IQ Server may need some time to process them. You can check the status and obtain results using the following GET REST Resource:

```
GET /api/v2/evaluation/applications/{applicationInternalId}/results/{resultId}
```

In our example, we're going to use the resultsURL and as before, cURL:

```
curl -u admin:admin123 -X GET 'http://localhost:8070/api/v2/evaluation/applications/
529b7f71bb714eca8955e5d66687ae2c/results/917e0c5e92a646a5b8879a40890078bc'
```

If the report is ready, you will receive results similar to these below:

If the report is not ready, you will receive a (404) error.


```
{
   "submittedDate":"2015-02-13T18:01:42.082-05:00",
   "evaluationDate":"2015-02-13T18:01:42.084-05:00",
   "applicationId":"529b7f71bb714eca8955e5d66687ae2c",
   "results":[
      {
         "component":{
            "hash":null,
            "componentIdentifier":{
               "format":"maven",
               "coordinates":{
                  "artifactId":"commons-fileupload",
                  "extension":"jar",
                  "groupId":"commons-fileupload",
                  "version":"1.2.2"
               }
            },
            "proprietary":false
         },
         "matchState":"exact",
         "catalogDate":"2010-07-29T16:41:12.000-04:00",
         "licenseData":{
            "declaredLicenses":[
               {
                  "licenseId":"Apache-2.0",
                  "licenseName":"Apache-2.0"
               }
            ],
            "observedLicenses":[
               {
                  "licenseId":"Apache-2.0",
                  "licenseName":"Apache-2.0"
               }
            ],
            "overriddenLicenses":[

            ],
            "status":"Open"
         },
         "securityData":{
```

```
      "securityIssues":[
         {
            "source":"cve",
            "reference":"CVE-2013-2186",
            "severity":7.5,
            "status":"Open",
            "url":"http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2013-2186",
            "threatCategory":"critical"
         },
         {
            "source":"osvdb",
            "reference":"98703",
            "severity":7.5,
            "status":"Open",
            "url":"http://osvdb.org/98703",
            "threatCategory":"critical"
         },
         {
            "source":"cve",
            "reference":"CVE-2014-0050",
            "severity":5.0,
            "status":"Open",
            "url":"http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-0050",
            "threatCategory":"severe"
         },
         {
            "source":"osvdb",
            "reference":"102945",
            "severity":5.0,
            "status":"Open",
            "url":"http://osvdb.org/102945",
            "threatCategory":"severe"
         }
      ]
   },
   "policyData":{
      "policyViolations":[
         {
            "policyId":"0ccc3321662d491c86b32554ea84e4e9",
            "policyName":"Security-High",
            "threatLevel":9,
            "constraintViolations":[
               {
                  "constraintId":"d2dfc91f3bf4460ba0da3f7201e4adb7",
                  "constraintName":"CVSS >=7 and <10",
                  "reasons":[
                     {
                        "reason":"Found 2 Security Vulnerabilities with Severity >= 7"
                     },
                     {
                        "reason":"Found 4 Security Vulnerabilities with Severity < 10"
                     },
                     {
                        "reason":"Found 4 Security Vulnerabilities with Status OPEN"
```

```
                        }
                     ]
                  }
               ]
            },
            {
               "policyId":"0f1b57e6788c4e9bb0f65cde3b6e9f1c",
               "policyName":"Security-Medium",
               "threatLevel":7,
               "constraintViolations":[
                  {
                     "constraintId":"12a71811de694876b51e4c06c10bad76",
                     "constraintName":"CVSS >=4 and <7",
                     "reasons":[
                        {
                           "reason":"Found 4 Security Vulnerabilities with Severity >= 4"
                        },
                        {
                           "reason":"Found 2 Security Vulnerabilities with Severity < 7"
                        },
                        {
                           "reason":"Found 4 Security Vulnerabilities with Status OPEN"
                        }
                     ]
                  }
               ]
            }
         ]
      }
   ],
   "isError":false,
   "errorMessage":null
}
```


The last bit of information indicates if there were any errors encountered during the evaluation. In this example, there were no errors.

Application REST APIs - v2

The primary functions of the Application REST APIs are creating, updating, and deleting applications. In addition to this, Application Categories (referred to as "tags") can be added and users / groups mapped to roles (permissions) for the application.

The currently available APIs include:

- GET - used to retrieve information, such as a list of organizations, their internal IDs, and their tags. This is also used for retrieving roles and mapping those roles to an application.
- POST - used to create applications.

- PUT - used for mapping roles to the application, as well as making any necessary changes to modifiable application properties (i.e. name, tags, or contact).
- DELETE - used to delete applications.

As mentioned in the introduction we will provide both the API, as well as an example using the HTTP client cURL. Additionally, to help demonstrate use of the APIs, we've approached this in a step-by-step manner that will start with gathering the necessary information to create and set permissions for an application.

If you already have an application, and wish to make an update, you can jump to the final step now.

Step 1 - Get the Organization ID

In order to create an application, it must have a parent organization, which means that you must have already created at least one organization in IQ Server. (The Organization REST API (see page 268) allows you to create organizations without using the GUI).

To retrieve an organization ID for an existing organization, we must start with our GET API call...

```
GET /api/v2/organizations
```

and find the organization ID. Additionally, any tags available to be applied to the application will be provided. To follow along using cURL, enter the following command:

```
curl -u admin:admin123 -X GET 'http://localhost:8070/api/v2/organizations'
```

This will produce a list of your organizations and associated information in a JSON format. Here is an example of what might be returned.


```
{
    "organizations": [
        {
            "id": "36d7e629462a4038b581488c347959bc",
            "name": "My Organization",
            "tags": [ ]
        },
        {
            "id": "f48b5344fa204a4c88df96b2d455d521",
            "name": "My Organization 2",
            "tags": [
                {
                    "id": "cd8fd2f4f289445b8975092e7d3045ba",
                    "name": "Distributed",
                    "description": "Applications that are provided for consumption outside the company",
```

```
                    "color": "yellow"
                },
                {
                    "id": "004d789684834f7c889c8b186a5ff24b",
                    "name": "Hosted",
                    "description": "Applications that are hosted such as services or software as a
service.",
                    "color": "grey"
                },
                {
                    "id": "da9e09887c754157a2113831ae7e99ac",
                    "name": "Internal",
                    "description": "Applications that are used only by your employees",
                    "color": "green"
                }
            ]
        }
    ]
}
```


Item	Description
id	This is the internal id for the organization.
name	This is the name of the organization, and is visible in the IQ Server GUI.
tags	Tags represent identifying characteristic for an application. Tags are created at the organization level, and then applied to applications. Policies can then select which tags, and in turn applications, the policy will be evaluated against.
id (tags)	the internal id for the tag. This will be used when creating the application.
name (tags)	the name of the tag, which is visible in the IQ Server GUI.
description (tags)	each tag is required to have a description. This description provides information related to the type of application it should be applied to. In many cases, you will have more than one organization, as does our example.

Step 2 - Create an Application

To create an application, you need several pieces of information. As we've already mentioned, you will need the organization ID, but you will also need:

Item	Description
publicId (application ID)	This is the application ID for the application. In the IQ Server GUI this is represented by the "Application" field. It must be unique.

name (application name)	This is the name of the application. In the IQ Server GUI this corresponds to the "Application Name" field. It must be unique.
contactUserName (application contact)	In the IQ Server GUI, this corresponds to the contact field for the application. The value entered here should represent the user name. If you are using LDAP and have configured it to work with the IQ Server, that user name can be used here.
tagId (internal tag ID)	As mentioned previously, a tag applied to an application will ensure it is evaluated by any policies that have selected the corresponding tag. The tag ID is returned as part of the organization information.

The application contact and tags are optional, but it is generally recommended to include them.

We'll be using the POST API call...

```
POST /api/v2/applications
```

and appending the information mentioned above into a JSON-formatted body. Here is an example of the body that's been formatted for easier review:

```
{
    "publicId": "MyApplicationID",
    "name": "MyFirstApplication",
    "organizationId": "f48b5344fa204a4c88df96b2d455d521",
    "contactUserName": "AppContact",
    "applicationTags": [
        {
            "tagId": "cd8fd2f4f289445b8975092e7d3045ba"
        }
    ]
}
```

Putting this all together, and using our cURL example, you should enter the following command:

```
curl -u admin:admin123 -X POST -H "Content-Type: application/json" -d '{"publicId":
"MyApplicationID","name":
"MyFirstApplication","organizationId":"f48b5344fa204a4c88df96b2d455d521","contactUserName":"AppContact","ap
plicationTags": [{"tagId":"cd8fd2f4f289445b8975092e7d3045ba"}]}' 'http://localhost:8070/api/v2/
applications'
```

If your application creation was successful, the system will respond with the following:

```
{
    "id": "4bb67dcfc86344e3a483832f8c496419",
    "publicId": "MyApplicationID",
    "name": "MyFirstApplication",
    "organizationId": "f48b5344fa204a4c88df96b2d455d521",
    "contactUserName": "AppContact",
    "applicationTags": [
        {
            "id": "9beee80c6fc148dfa51e8b0359ee4d4e",
            "tagId": "cd8fd2f4f289445b8975092e7d3045ba",
            "applicationId": "4bb67dcfc86344e3a483832f8c496419"
        }
    ]
}
```


Be sure to make note of the "applicationId". This is the internal application ID that will be used in the next steps when finding roles and mapping a user to them.

Step 3 - Find the Currently Available Roles

Before mapping any roles, it's a good idea to take a look at all the available roles for applications. This can be done using a simple GET:

```
GET /api/v2/applications/roles
```

This returns all the available roles for all applications:


```
{
    "roles": [
        {
            "id": "1da70fae1fd54d6cb7999871ebdb9a36",
            "name": "Developer",
            "description": "Allows to evaluate policies."
        },
        {
            "id": "1cddabf7fdaa47d6833454af10e0a3ef",
            "name": "Owner",
            "description": "Allows to manage policies."
        }
    ]
}
```


Item	Description
id	This is the internal ID for the role. There is no corresponding field in the IQ Server GUI.

name	This is the name of the role, which is exactly the same as the Role Name displayed in the IQ Server GUI. As we mentioned, there are two role names in the example above Developer and Owner.
description	This provides a description of the level of access the role grants.

As you may notice, there are two roles available for applications at this time. Before proceeding to the next step, take note of the **`roleId`** (in this example we will be adding an owner). This will be needed for mapping a user to the specific role.

Step 4 - Map Users to Roles

To map users to roles, we'll need the **`id`** (application ID) from our successful application creation, the **`roleId`** for the role returned above that you will be mapping a user to, and the following information:

Item	Description
type	The type of user, which at this time, can only be either USER or GROUP.
userOrGroupName	This is the username for the user you want to add to the role. If you are using LDAP and have configured it to work with the IQ Server, the LDAP user or group name can be used here.

For this step, we'll be using the following PUT API call…

```
PUT /api/v2/applications/{applicationInternalId}/roleMembers
```

with an appended JSON-formatted body. Here is an example of the body that's been formatted for easier review:

```
{
    "memberMappings": [
        {
            "roleId": "1cddabf7fdaa47d6833454af10e0a3ef",
            "members": [
                {
                    "type": "USER",
                    "userOrGroupName": "user-owner"
                }
            ]
        }
    ]
}
```

Putting this altogether, and using our cURL example, you should enter the following command:

```
curl -u admin:admin123 -X PUT -H "Content-Type: application/json" -d '{"memberMappings": [{"roleId":
"1da70fae1fd54d6cb7999871ebdb9a36","members": [{"type": "USER","userOrGroupName": "user-b"}]}]}' 'http://
localhost:8070/api/v2/applications/4bb67dcfc86344e3a483832f8c496419/roleMembers'
```

As before, standard HTTP response codes will be returned.

ⓘ You can map multiple roles in a single PUT.

Step 5 - Update Application Information

This final step isn't a requirement to complete an application, but it will be necessary for any application you want to update.

Before updating the application, you will need the id for the application. If you haven't recorded this, you can obtain it by using the **`publicId`**. To retrieve the information for the application use the following GET API call...

```
GET /api/v2/applications?publicId={YourPublicId}
```

ⓘ All applications can always be returned by omitting the reference to the public id.

Using the cURL command, it would look like this:

```
curl -u admin:admin123 -X GET 'http://localhost:8070/api/v2/applications?publicId=MyApplicationID'
```

If you like, you can also return multiple applications by appending additional **`&publicId={SomeOtherPublicId}`** to the command above.

Placing your application's publicId where **`{YourPublicId}`** is, the following information will be returned (unique to your application). This has been formatted for readability:

```
{
    "applications": [
        {
            "id": "4bb67dcfc86344e3a483832f8c496419",
            "publicId": "MyApplicationID",
            "name": "MySecondApplication",
            "organizationId": "bb41817bd3e2403a8a52fe8bcd8fe25a",
            "contactUserName": "NewAppContact",
            "applicationTags": [
                {
```

```
                    "id": "9beee80c6fc148dfa51e8b0359ee4d4e",
                    "tagId": "cfea8fa79df64283bd64e5b6b624ba48",
                    "applicationId": "4bb67dcfc86344e3a483832f8c496419"
                }
            ]
        }
    ]
}
```


From the information returned above, make note of the following:

- "id"
- "organizationId"

The information for an application that can be updated, follows the same rules as editing an application via the IQ Server GUI. In addition, you will also need any information you will be editing, which can include any or all of the following:

- "publicId" (see Important note below)
- "name"
- "contactUserName"
- "applicationTags" (tagID)

If you are looking to update the role mapping for the application, simply follow the instructions from step 4.

Now, you should be ready to apply your changes. Before we move on to the command, let's take a look at a sample body for the attached JSON file:

```
{
    "id": "4bb67dcfc86344e3a483832f8c496419",
    "publicId": "MyApplicationID",
    "name": "MySecondApplication",
    "organizationId": "bb41817bd3e2403a8a52fe8bcd8fe25a",
    "contactUserName": "NewAppContact",
    "applicationTags": [
        {
            "tagId": "cfea8fa79df64283bd64e5b6b624ba48"
        }
    ]
}
```

Using cURL enter the following command:

```
curl -u admin:admin123 -X PUT -H "Content-Type: application/json" -d
'{"id":"4bb67dcfc86344e3a483832f8c496419","publicId":"MyApplicationID","name":"MySecondApplication","organi
```

```
zationId":"bb41817bd3e2403a8a52fe8bcd8fe25a","contactUserName":"NewAppContact","applicationTags":
[{"tagId":"cfea8fa79df64283bd64e5b6b624ba48"}]}' 'http://localhost:8070/api/v2/applications/
4bb67dcfc86344e3a483832f8c496419'
```

If your update is successful, you will see something similar to the formatted JSON file below.

```
{
    "id": "4bb67dcfc86344e3a483832f8c496419",
    "publicId": "MyApplicationID",
    "name": "MySecondApplication",
    "organizationId": "bb41817bd3e2403a8a52fe8bcd8fe25a",
    "contactUserName": "NewAppContact",
    "applicationTags": [
        {
            "id": "9beee80c6fc148dfa51e8b0359ee4d4e",
            "tagId": "cfea8fa79df64283bd64e5b6b624ba48",
            "applicationId": "4bb67dcfc86344e3a483832f8c496419"
        }
    ]
}
```

If you change an application PublicId, remember to update your integration tools to use the new value.

Deleting an Application

In case you wish to delete an application permanently, the Application REST APIs offer the ability to do so.

Deleting an application is a destructive action that will permanently remove an application and all data associated with it. Ensure first that you are deleting the correct application and that deleting the application will not adversely affect any integrations that depend on the presence of the application.

Before deleting the application, you will need the id for the application. See updating an application for more information on how to obtain the application id using the application publicId.

The following DELETE API call is available for deleting an application…

```
DELETE /api/v2/applications/{applicationInternalId}
```

Using cURL:

```
curl -u admin:admin123 -X DELETE 'http://localhost:8070/api/v2/applications/
4bb67dcfc86344e3a483832f8c496419'
```

Standard HTTP status codes will be returned in response to a request to delete an application.

Violation REST API - v2

The Policy Violation REST APIs allow you to access and extract policy violations gathered during the evaluation of applications. In most cases the desire for getting to this data is to integrate into other tools your company may have. For example you may have a specific dashboard or reporting application that should have this data.

Whatever the case, just as with the other REST APIs, this is all done using REST API calls. For accessing policy violation information the following API is used:

GET	Used to retrieve policy information, such as a list of policy ids as well as a list of violations based on a specific Policy ID, or list of IDs.

As mentioned previously, we will provide both the API, as well as examples using the HTTP client cURL. This is only for demonstration purposes and displaying the necessary input, and desired output.

Additionally, to help demonstrate this, we've approached this in a step-by-step manner that will start with gathering policy ids, and then requesting the violations.

Before You Get Started

As with the other REST APIs, you will need a username and password to interface with the IQ Server. In addition, because access to this data is granted based on the roles (permissions) you have set up, you may wish to create one specifically for this process.

Other than this, the only piece you may need in order to follow along with our instructions is cURL, or a comparable HTTP client.

Step 1 - Get the Policy IDs

To access policy violation information you need the Policy ID(s). For this reason, we start with the GET API call...

```
GET /api/v2/policies/
```

which will return a list of all Policy IDs. To follow along using cURL, enter the following command:

```
curl -u admin:admin123 -X GET 'http://localhost:8070/api/v2/policies'
```

The action above will produce a list of your policies in a JSON format. Here is an example of what might be returned.


```
{
    "policies": [
        {
            "id": "6984017845c645b0ad0c95401ad4f17d",
            "name": "My Application Policy",
            "ownerId": "36d7e629462a4038b581488c347959bc",
            "ownerType": "APPLICATION",
            "threatLevel": 5,
            "policyType": "quality"
        },
    ]
}
```


As you can see above, we've used the admin user which is shipped with the IQ Server, as well as the default IQ Server location. The user you use may differ depending on your configuration.

Item	Description
id	This is the internal id for the policy.
name	This is the name of the policy, and is visible in the IQ Server GUI.
ownerId	This is the internal id for the organization or application that the policy resides in, and is not visible within the IQ Server GUI.
ownerType	This indicates whether the policy is for an organization or application.
threatLevel	This is the threat level that is set for the policy.
policyType	This is the policy type, which is based on the conditions used in the policy.

In many cases, you will have many policies, especially if you are retrieving information for an account that has access to many applications and/or organization.

Step 2 - Get the Policy Violations

Now that you have the Policy IDs, they can be used to gather a list of policy violations. To do this, you will need the Policy IDs you retrieved from step one. For example:

```
"id": "6984017845c645b0ad0c95401ad4f17d"
```

> Policy IDs are unique, and what is used above is just an example.

Slightly different from before, we will use the GET API call…

```
GET /api/v2/policyViolations?p=policyid1
```

which passes a simple query for Policy IDs. For each policy we want to retrieve violations for, we will include that ID. If desired we can retrieve violations for multiple Policy IDs. To do this, just make sure you add &p="The Policy ID" for each policy you want included. Here is an example of the API with the Policy ID we retrieved:

```
GET /api/v2/policyViolations?p=6984017845c645b0ad0c95401ad4f17d
```

Putting this all together, and using our cURL example, you should enter the following command:

```
curl -u admin:admin123 -X GET 'http://localhost:8070/api/v2/policyViolations?
p=6984017845c645b0ad0c95401ad4f17d'
```

If your query was successful, the system will respond with something like this:


```
{
   "applicationViolations":[
      {
         "application":{
            "id":"529b7f71bb714eca8955e5d66687ae2c",
            "publicId":"MyAppID1",
            "name":"MyApplications",
            "organizationId":"36d7e629462a4038b581488c347959bc",
            "contactUserName":null
         },
         "policyViolations":[
            {
```

```
            "policyId":"6984017845c645b0ad0c95401ad4f17d",
            "policyName":"Security-High",
            "stageId":"build",
            "reportUrl":"ui/links/application/MyAppID1/report/c0ddef
            c4512f42d0bcbe29029e2be117",
            "threatLevel":9,
            "constraintViolations":[
               {
                  "constraintId":"19011de290b147a38c820ad7bd5c653d",
                  "constraintName":"CVSS >=7 and <10",
                  "reasons":[
                     {
                        "reason":"Found 2 Security Vulnerabilities with Severity >= 7"
                     },
                     {
                        "reason":"Found 4 Security Vulnerabilities with Severity < 10"
                     },
                     {
                        "reason":"Found 4 Security Vulnerabilities with Status OPEN"
                     }
                  ]
               }
            ],
            "component":{
               "hash":"384faa82e193d4e4b054",
               "componentIdentifier":{
                  "format":"maven",
                  "coordinates":{
                     "artifactId":"tomcat-util",
                     "classifier":"",
                     "extension":"jar",
                     "groupId":"tomcat",
                     "version":"5.5.23"
                  }
               },
               "proprietary":false
            }
         }
      ]
   }
]
}
```


And there you have it, you've just retrieved policy violations. In the table below, each of the categories of data that is returned, as well as each field, are described:

Item	Descrption
application	Category containing specific information about the application.
id	The internal id.

publicId	The application ID. In the IQ Server GUI this is represented by the "Application" field.
name	The name of the application. In the IQ Server GUI this corresponds to the "Application Name" field.
organizationId	The internal id for the organization that the application resides in, and is not visible within the IQ Server GUI.
contactUserName	This is typically the person in charge of the application. In the IQ Server GUI, it corresponds to the contact field for the application.
policyViolations	A subcategory of the application, and provides specific information about the policy and corresponding violations that were found.
policyId	The internal id for the policy.
policyName	The name of the policy, and is visible in the IQ Server GUI.
stageId	Stage in which the policy violation occurred in. It is displayed in various places within the IQ Server GUI, including the associated Application Composition Report.
reportUrl	This is the URL to the Application Composition Report associated with the evaluation that found the listed policy violations.
threatLevel	This the threat level of the policy that was violated.
constraintViolations	This is a subcategory for Policy Violations, and includes all information related to specific constraint that was violated.
constraintId	This is the internal id for the constraint, and is not visible in the IQ Server GUI, or in the associated Application Composition Report.
constraintName	This is the name of the constraint and is visible in the policy area where the policy was created (i.e either the organization or application). It is also displayed in the Application Composition Report and various tools that connect to the IQ Server.
reasons	This is a subcategory of Constraint Violations, and gives the reason why the violation occurred.
reason	The reason is formed by the value(s) for the condition(s) violated. Conditions are visible where the policy was created (i.e either the organization or application). It is also displayed in the Application Composition Report and various tools that connect to the IQ Server.

component	Component is a subcategory of Policy Violations, and includes information about the component(s) causing the violation to occur. ⓘ The component field only refers to a single component. If another component violates the same policy, another entry for that in "policyViolations" would be present.
hash	This is the hash value for the component. For example, in the case of Java components, this would be matched to a Java repository (e.g. Central). If you have proprietary components configured, it would be matched against your list of proprietary components.
componentIdentifier	This is simply a container for the component information. It will always include the format and the coordinates.
format	This is the format the component is in, and will determine what type of coordinate information is displayed.
coordinates	This will depend on the format. An example would be Maven, which uses a G : A : E : C : V (Group, Artifact Id, Extension, Classifier, and Version) for the component. In this example, the fields provided are: component.

Report-related REST APIs - v2

The Application Composition Report (see page 193) is created when an evaluation occurs. There are two available REST APIs associated with obtaining information related to these reports:

Reports URL REST API (v2)

The Reports API provides a summary of an application's most recent reports across the various stages (e.g. Build, Stage Release, and Release). The API consists of using the GET REST resource that is part of both the applications and reports APIs.

To assist in learning to use this API we will provide both the API as well as an example using the HTTP client cURL. We've approached this in a step-by-step manner that will start with gathering the internal application ID and then retrieving the report information.

In our example below, we have isolated retrieving a report for a single application. However, if desired the reports for all applications can also be retrieved. In this case, skip to Step 2, where a corresponding tip has been included to assist.

Step 1 - Get the Application ID

First, you will use the application's public ID to retrieve the internal application ID. This is done using the following GET REST resource from our application API…

```
GET /api/v2/applications?publicId={YourPublicId}
```

All applications can always be returned by omitting the reference to the public ID.

Using the cURL command, it would look like this:

```
curl -u admin:admin123 -X GET 'http://localhost:8070/api/v2/applications?publicId=MyApplicationID'
```

If you like, you can also return multiple applications by appending additional **`&publicId={SomeOtherPublicId}`** to the command above.

Information will be returned (unique to your application). This has been formatted for readability:


```
{
    "applications": [
        {
            "id": "4537e6fe68c24dd5ac83efd97d4fc2f4",
            "publicId": "MyApplicationID",
            "name": "MyApplication",
            "organizationId": "bb41817bd3e2403a8a52fe8bcd8fe25a",
            "contactUserName": "NewAppContact",
            "applicationTags": [
                {
                    "id": "9beee80c6fc148dfa51e8b0359ee4d4e",
                    "tagId": "cfea8fa79df64283bd64e5b6b624ba48",
                    "applicationId": "4bb67dcfc86344e3a483832f8c496419"
                }
            ]
        }
    ]
}
```


From the information returned above, make note of the id. This is the internal application ID.

The public ID can be found via the IQ Server GUI by navigating to the respective application and copying the application ID located just below the application name.

Step 2 - Get Report URLs

Next we will take the internal application ID from Step 1 and use that with the GET REST resource for the Reports API. This is done with our GET REST resource associated with our Reports API…

```
GET /api/v2/reports/applications/{applicationInternalid}
```

If you would like to obtain the report information for all applications, simply leave off the "id" and all report information for all applications will be provided.

Using cURL, it will look like this…

```
curl -u admin:admin123 -X GET 'http://localhost:8070/api/v2/reports/applications/
4bb67dcfc86344e3a483832f8c496419'
```

If any report information is found for the application, you will receive JSON formatted data. An example is included below.


```
[
    {
        "stage": "build",
        "applicationId": "4537e6fe68c24dd5ac83efd97d4fc2f4",
        "evaluationDate": "2015-01-16T13:14:32.139-05:00",
        "reportHtmlUrl": "ui/links/application/Test123/report/474ca07881554f8fbec168ec25
        d9616a",
        "embeddableReportHtmlUrl": "ui/links/application/Test123/report/474ca07881554f8fbec168ec25
        d9616a/embeddable",
        "reportPdfUrl": "ui/links/application/Test123/report/474ca07881554f8fbec168ec25
        d9616a/pdf",
        "reportDataUrl": "api/v2/applications/Test123/reports/474ca07881554f8fbec168ec25
        d9616a"
    },
    {
        "stage": "stage-release",
        "applicationId": "4537e6fe68c24dd5ac83efd97d4fc2f4",
        "evaluationDate": "2014-07-14T15:08:09.501-04:00",
        "reportHtmlUrl": "ui/links/application/Test123/report/2dbfd514e5ad497598086c3e4c
        89c413",
        "embeddableReportHtmlUrl": "ui/links/application/Test123/report/2dbfd514e5ad497598086c3e4c
        89c413/embeddable",
```

```
        "reportPdfUrl": "ui/links/application/Test123/report/2dbfd514e5ad497598086c3e4c
        89c413/pdf",
        "reportDataUrl": "api/v2/applications/Test123/reports/2dbfd514e5ad497598086c3e4c
        89c413"
    },
    {
        "stage": "release",
        "applicationId": "4537e6fe68c24dd5ac83efd97d4fc2f4",
        "evaluationDate": "2014-08-31T12:01:46.179-04:00",
        "reportHtmlUrl": "ui/links/application/Test123/report/97cccbeeb58d4aac83264993ef
        6bf4d8",
        "embeddableReportHtmlUrl": "ui/links/application/Test123/report/97cccbeeb58d4aac83264993ef
        6bf4d8/embeddable",
        "reportPdfUrl": "ui/links/application/Test123/report/97cccbeeb58d4aac83264993ef
        6bf4d8/pdf",
        "reportDataUrl": "api/v2/applications/Test123/reports/97cccbeeb58d4aac83264993ef
        6bf4d8"
    }
]'
```


Component Details by Report REST API (v2)

When an application is evaluated, information found during the evaluation is provided via the Application Composition Report. While this is the easiest way to review this information, it can also be exported to a JSON file using the Component Information API.

The API works by sending a REST request to the IQ Server. This request involves using a specially formatted URL and any HTTP client. In the example we have provided we make use of cURL and the command line. In addition, we format the JSON to make it a bit more readable.

Step 1 - Sending the Request

First, let's take a look at the GET API we'll be using:

```
GET http://localhost:8070/api/v2/applications/[applicationPublicId]/reports/[reportId]
```

As you may have noticed, this API uses a URL specific to the location of the report. There are two pieces of information you will need in order to retrieve the results.

- applicationId - The application ID for the specific application.
- applicationPublicId - The public ID of the specific application.
- reportId - The ID of the specific report.

There are a variety of ways to retrieve this information, including gathering it by using the Component Search API (see page 236). However, in our example, we're just going to pull this information from the log output.

```
...

************************************************************************************
[INFO] Policy Action: None
[INFO] CLM stage: build
[INFO] Summary of policy violations: 6 critical, 11 severe, 1 moderate
[INFO] The detailed report can be viewed online at http://localhost:8070/ui/links/application/MyApp-1234/
report/68b6bdb1573a40eeb4205d890b602525
[INFO]
************************************************************************************
```

We'll use the link for the report to gather the information we need:

```
http://localhost:8070/ui/links/application/MyApp-1234/report/68b6bdb1573a40eeb4205d890b602525
```

In the example above, the section of the URL following "application" and before the section "report", is the application's public ID. Similarly, the section after "report" is the report ID.

> The report is also presented in a full GUI. To access this, simply paste the link in your browser. However, you will need to be at least a member of the developer group for the application that has been evaluated, or you will not be able to access the report.

Now, we can download the data using our HTTP request tool.

Step 2 - Downloading Component Information

Again, in our example we are going to use cURL, though any HTTP client could be used. Here is what our request looks like:

```
curl -u admin:admin123 -X GET "http://localhost:8070/api/v2/applications/MyApp-1234/reports/
68b6bdb1573a40eeb4205d890b602525"
```

> Included in our cURL example is the default username and password for the admin account. Your account credentials may vary, but are necessary in order for the request to be processed. If the username and password provided are not at least within the developer role for the application, an error will be returned.

Looking at the result through a JSON Viewer, you should see something like this:

```
{
```

```
   "components":[
      {
         "hash":"1249e25aebb15358bedd",
         "componentIdentifier":{
            "format":"maven",
            "coordinates":{
               "artifactId":"tomcat-util",
               "groupId":"tomcat",
               "version":"5.5.23"
                           "extension":"jar",
                           "classifier":""
            }
         },
         "proprietary":false,
         "matchState":"exact",
         "pathnames":[
            "sample-application.zip/tomcat-util-5.5.23.jar"
         ],
         "licenseData":{
            "declaredLicenses":[
               {
                  "licenseId":"Apache-2.0",
                  "licenseName":"Apache-2.0"
               }
            ],
            "observedLicenses":[
               {
                  "licenseId":"No-Sources",
                  "licenseName":"No Sources"
               }
            ],
            "overriddenLicenses":[

            ],
            "status":"Open"
         },
         "securityData":{
            "securityIssues":[
               {
                  "source":"cve",
                  "reference":"CVE-2007-3385",
                  "severity":4.3,
                  "status":"Open",
                  "url":"http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2007-3385",
                  "threatCategory":"severe"
               }
            ]
         }
      }
   ]
}
```


Now that you have access to this data, it can be packaged in a variety of ways. For example, you may want to use this as a bill of materials to include associated license data.

In cases where you want a version of the report without any of the associated IQ Server navigation elements (e.g. for use in a custom tool), use the embeddableReportHtmlUrl.

Accessing REST APIs via Reverse Proxy Authentication

When authentication is handled by a reverse proxy server as described in the section Reverse Proxy Authentication (see page 96), API requests that change data, i.e. POST, PUT and DELETE requests, are subject to cross-site request forgery (CSRF) protection. For these requests to be accepted by IQ Server, they need to include the HTTP header **`X-CSRF-TOKEN`** along with an HTTP cookie named **`CLM-CSRF-TOKEN`** where both the header and the cookie carry the same value. The specific value chosen is irrelevant, it only needs to be the same for the header and the cookie.

Please refer to the documentation of your respective HTTP client on how to supply the header and cookie. For the cURL tool used in our earlier examples, this can be accomplished as follows:

```
curl --header "X-CSRF-TOKEN: api" --cookie "CLM-CSRF-TOKEN=api" ...
```

Organization REST APIs - v2

The Organization REST API allows you to create new organizations, browse the list of organizations, and map users / groups to roles (permissions) for an organization.

Supported HTTP methods include:

- GET - used to retrieve the list of organizations (including their internal IDs and associated tags) and for retrieving roles and their members for an organization.
- POST - used to create organizations.
- PUT - used for mapping members to roles for an organization.

Supported operations include:

Add Organization

Creating an organization is quite simple, only requiring a unique name:

Item	Description
name	The name of the new organization. In the IQ Server GUI this corresponds to the "Organization Name" field. It must be unique.

We'll be making a POST API call for this operation:

```
POST /api/v2/organizations
```

The JSON payload contains the `name` parameter, for example:

```
{
    "name": "My Organization"
}
```

You can run the following cURL command to make the request:

```
curl -u admin:admin123 -X POST -H "Content-Type: application/json" -d '{"name": "My Organization"}'
'http://localhost:8070/api/v2/organizations'
```

If your organization was successfully created, the system will respond with the following:

```
{
    "id": "ab3d35233046480a9c30bb2437a48103",
    "name": "My Organization",
    "tags": []
}
```

Item	Description
id	This is the internal id for the organization.
name	This is the name of the organization, and is visible in the IQ Server GUI.
tags	Tags are created at the organization level, and then applied to applications. A newly-created organization will have no tags.

Note that the "id" returned is the ID of the new organization. This can be used in other operations that require an organization ID.

Get All Organizations

To get a list of the organizations we must start with a GET call:

```
GET /api/v2/organizations
```

We can use cURL to make the request:

```
curl -u admin:admin123 -X GET 'http://localhost:8070/api/v2/organizations'
```

This will produce a list of your organizations and associated information in a JSON format. Here is an example of what might be returned.

```
{
    "organizations": [
        {
            "id": "36d7e629462a4038b581488c347959bc",
            "name": "My Organization",
            "tags": [ ]
        },
        {
            "id": "f48b5344fa204a4c88df96b2d455d521",
            "name": "My Organization 2",
            "tags": [
                {
                    "id": "cd8fd2f4f289445b8975092e7d3045ba",
                    "name": "Distributed",
                    "description": "Applications that are provided for consumption outside the company",
                    "color": "yellow"
                },
                {
                    "id": "004d789684834f7c889c8b186a5ff24b",
                    "name": "Hosted",
                    "description": "Applications that are hosted such as services or software as a
service.",
                    "color": "grey"
                },
                {
                    "id": "da9e09887c754157a2113831ae7e99ac",
                    "name": "Internal",
                    "description": "Applications that are used only by your employees",
                    "color": "green"
                }
            ]
        }
    ]
}
```

Item	Description
id	This is the internal id for the organization.
name	This is the name of the organization, and is visible in the IQ Server GUI.
tags	Tags represent identifying characteristics for an application. Tags are created at the organization level, and then applied to applications. Policies can then select which tags, and in turn applications, the policy will be evaluated against.
id (tags)	The internal id for the tag. This will be used when creating the application.
name (tags)	The name of the tag, which is visible in the IQ Server GUI.
description (tags)	Each tag is required to have a description. This description provides information related to the type of application it should be applied to.

Find the Currently Available Roles

Before mapping any roles, it's a good idea to take a look at all the available roles for organizations. This can be done using a simple GET which is equivalent to that used for applications i.e.

```
GET /api/v2/applications/roles
```

This returns all the available roles for all organizations:

```
{
    "roles": [
        {
            "id": "2cb71b3468d649789163ea2e212b541e",
            "name": "Application Evaluator",
            "description": "Evaluates applications and views policy violation summary
results."
        },
        {
            "id": "90c7c98683b4471cb77a916744540bcc",
            "name": "Component Evaluator",
            "description": "Evaluates individual components and views policy violation
results for a specified application."
        },
       {
           "id": "1da70fae1fd54d6cb7999871ebdb9a36",
           "name": "Developer",
           "description": "Views all information for their assigned organization or
application."
       },
       {
           "id": "1cddabf7fdaa47d6833454af10e0a3ef",
           "name": "Owner",
           "description": "Manages assigned organizations, applications, policies, and policy
violations."
       }
    ]
}
```


Item	Description
id	This is the internal ID for the role. There is no corresponding field in the IQ Server GUI.
name	This is the name of the role, which is exactly the same as the Role Name displayed in the IQ Server GUI.
description	This provides a description of the level of access the role grants.

As you may notice, there are four roles available for organizations at this time. Before proceeding to the next step, take note of the role **`id`** (in this example we will be adding an owner). This will be needed for mapping a user to the specific role.

Map Users to Roles

To map users to roles, we'll need the **`id`** (organization ID) from our successful organization creation, the role **`id`** for the role returned above that you will be mapping a user to, and the following information:

Item	Description
type	The type of user, which at this time, can only be either USER or GROUP.
userOrGroupName	This is the username for the user you want to add to the role. If you are using LDAP and have configured it to work with the IQ Server, the LDAP user or group name can be used here.

For this step, we'll be using the following PUT API call…

```
PUT /api/v2/organizations/{organizationInternalId}/roleMembers
```

with an appended JSON-formatted body. Here is an example of the body that's been formatted for easier review:

```
{
    "memberMappings": [
        {
            "roleId": "1cddabf7fdaa47d6833454af10e0a3ef",
            "members": [
                {
                    "type": "USER",
                    "userOrGroupName": "user-owner"
                }
            ]
        }
    ]
}
```

Putting this altogether, and using our cURL example, you should enter the following command:

```
curl -u admin:admin123 -X PUT -H "Content-Type: application/json" -d '{"memberMappings": [{"roleId":
"1cddabf7fdaa47d6833454af10e0a3ef","members": [{"type": "USER","userOrGroupName": "user-owner"}]}]}'
'http://localhost:8070/api/v2/organizations/ab3d35233046480a9c30bb2437a48103/roleMembers'
```

As before, standard HTTP response codes will be returned.

ⓘ You can map multiple roles in a single PUT.

Get All Role Members

To get the members of each role for an organization we can use this GET call:

```
GET /api/v2/organizations/{organizationInternalId}/roleMembers
```

We can use cURL to make the request:

```
curl -u admin:admin123 -X GET 'http://localhost:8070/api/v2/organizations/ab3d35233046480a9c30bb2437a48103/
roleMembers'
```

This will produce the mappings from all roles to their members for your organization in a JSON format. Here is an example of what might be returned.

```
{
        "memberMappings": [{
                "roleId": "2cb71b3468d649789163ea2e212b541e",
                "members": [{
                        "type": "GROUP",
                        "userOrGroupName": "(all-authenticated-users)"
                }]
        },
        ...
        {
                "roleId": "1cddabf7fdaa47d6833454af10e0a3ef",
                "members": [{
                        "type": "USER",
                        "userOrGroupName": "user-owner"
                }]
        }]
}
```

Component Versions REST API - v2

The Component Versions API returns a list of versions for a component. This API method is available via a POST resource:

```
POST api/v2/components/versions
```

In order to use the API you must know the component identifier for the component you wish to find the versions for. A non-exhaustive list of examples for various formats is provided.

Maven Example

When finding a version for a Maven component, an appropriate Maven component identifier must be used.

```
{
        "format": "maven",
        "coordinates": {
                "artifactId": "tomcat-util",
                "groupId": "tomcat"
        }
}
```

You can use cURL to run this request.

```
curl -u admin:admin123 -X POST -H "Content-Type: application/json" -d '{ "format": "maven", "coordinates":
{ "artifactId": "tomcat-util", "groupId": "tomcat" } }' 'http://localhost:8070/api/v2/components/versions'
```

Javascript Example

For Javascript components the a-name (authoritative name) must be used.

```
{
        "format": "a-name",
        "coordinates": {
                "name": "vizion",
        "qualifier" : ""
        }
}
```

You can also use cURL for this request.

```
curl -u admin:admin123 -X POST -H "Content-Type: application/json" -d '{ "format": "a-name", "coordinates":
{ "name": "vizion", "qualifier": "" } }' 'http://localhost:8070/api/v2/components/versions'
```

Python Example

For Python components a component identifier appropriate for the PyPI format must be utilized.

```
{
        "format": "pypi",
        "coordinates": {
```

```
                "name": "pyOpenSSL"
        }
}
```

This also has a corresponding cURL request.

```
 curl -u admin:admin123 -X POST -H "Content-Type: application/json" -d '{ "format": "pypi", "coordinates": { "name": "pyOpenSSL" } }' 'http://localhost:8070/api/v2/components/versions'
```

Response

You will receive a response containing a JSON array with the known versions sorted in ascending order. For the Maven example above, you will see something like the following.

```
[
  "3.2.1",
  "3.3.2",
  "4.0.6",
  "4.1.31",
  "4.1.34",
  "4.1.36",
  "5.0.16",
  "5.0.18",
  "5.0.28",
  "5.5.4",
  "5.5.7-alpha",
  "5.5.7",
  "5.5.8-alpha",
  "5.5.9-alpha",
  "5.5.9",
  "5.5.12",
  "5.5.15",
  "5.5.23"
]
```

Component Labels REST API - v2

The Component Labels API adds or removes labels for components of an application. The API was Introduced in IQ Server 1.48.

Adding Application Component Labels

Adding an application component label requires the component hash, label name, and internal application ID for the component label.

Construct a POST REST request with the following path:

- componentHash - A sha1 hash of the component, which can be found in the source repository.
- labelName - The name of an existing label that can be applied to the application.
- applicationId - The internal ID of the application. Obtained from an 'applications' REST API call with: `GET /api/v2/applications?publicId={YourPublicId}`.

```
POST /api/v2/components/{componentHash}/labels/{labelName}/applications/
{applicationId}
```

The server will respond with a 204 status upon success.

Example Adding Application Component Labels

```
curl -u admin:admin123 -X POST 'http://localhost:8070/api/v2/components/
1249e25aebb15358bedd/labels/Architecture-Deprecated/applications/
5065550c24514129bf66f9afd5c7be60'
```

Deleting Application Component Labels

Deleting an application component label requires the component hash, label name, and internal application ID for the component label.

Construct a DELETE REST request with the following path:

- componentHash - A sha1 hash of the component, which can be found in the source repository.
- labelName - The name of an existing label that can be applied to the application.
- applicationId - The internal ID of the application. Obtained from an 'applications' REST API call.

```
DELETE /api/v2/components/{componentHash}/labels/{labelName}/applications/
{applicationId}
```

The server will respond with a 204 status upon success.

Example Deleting Application Component Labels

```
curl -u admin:admin123 -X DELETE 'http://localhost:8070/api/v2/components/
1249e25aebb15358bedd/labels/Architecture-Deprecated/applications/
5065550c24514129bf66f9afd5c7be60'
```

IQ Server Webhooks

Overview

Webhooks are HTTP callbacks that POST data to defined URLs. They let you build integrations registered to certain events on IQ Server. When a registered event is triggered, an HTTP POST payload is sent to the webhook's defined URL.

Use webhooks to receive notifications about events that happen in IQ Server. When an event occurs - for example, when an application evaluation is completed - IQ Server creates an event object. This object has relevant information about what just happened, such as the type of event and any data associated with the event. IQ Server then sends the event object to defined URLs using HTTP POST requests.

IQ Server lets you use webhooks for policy management, application evaluation, security vulnerability override management, and license override management events.

Each webhook is defined by the following:

- The URL to POST the payload.
- An optional secret key that ensures authenticity of the source.
- Event types subscribed by the webhook.

Configuring Webhooks

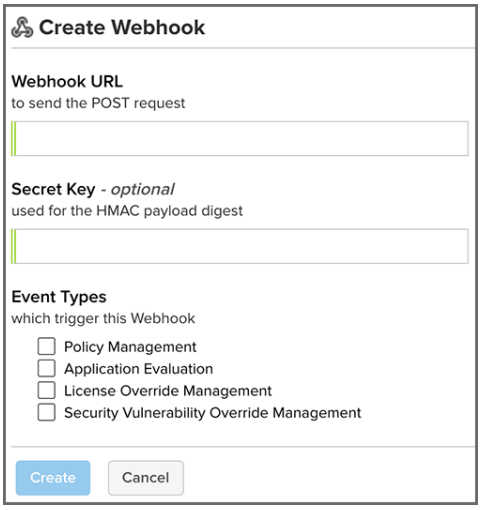
Webhooks can only be created by a System Administrator. Webhooks do not have a permissions model and will send HTTP requests for every event in the system. Be aware that the system consuming the webhooks will have access to all the data provided by the event types.

Creating Webhooks

To create a webhook, perform the following steps:

1. Click the *System Preferences* icon  in the IQ Server toolbar and then click *Webhooks*.

2. Click the ***Create Webhook*** button. The *Create Webhook* dialog opens:



3. Enter the webhook URL.
4. Optional: Enter the webhook secret key.
5. Select one or more Event Types.
6. Click *Create*.

Editing Webhooks

To edit a webhook, perform the following steps:

1. Click the *System Preferences* icon in the IQ Server toolbar and then click *Webhooks*.
2. Click the webhook you want to edit.
3. Edit the URL, secret key, or selected event types and then click *Update*.

Deleting Webhooks

To delete a webhook, perform the following steps:

1. Click the *System Preferences* icon in the IQ Server toolbar and then click *Webhooks*.
2. Click the webhook you want to delete.
3. Click *Remove Webhook*.
4. Click *Continue* in the *Remove Webhook* dialog.

Working with HMAC Payloads

If you enable a secret key to generate an HMAC digest, a special header is sent with all of your webhook payloads. This header is **`X-Nexus-Webhook-Signature`** and ensures that you receive an authentic message.

Webhooks can be consumed easily in node.js. Use the following setup to get started, substituting *foo* for the secret key you configured with your webhook:

Setup in terminal

```
npm init
npm install express
npm install body-parser
echo {\"secretKey\":\"foo\"} > settings.json
```

When verifying the HMAC digest, the HmacDigest value should match the signature value.

Add the code below to a file named `app.js`, and run the web hook listener via the command: `node app.js`

Example Webhook Consumer

```
const express = require('express');
const app = express();
const bodyParser = require('body-parser');
const settings = require('./settings.json');
const crypto = require('crypto');

app.use(bodyParser.json());

app.post('/', function(req, res) {
  const body = req.body;
  const signature = req.headers['x-nexus-webhook-signature'];
  var hmacDigest = crypto.createHmac("sha1",
settings.secretKey).update(JSON.stringify(body)).digest("hex");

  console.log('Webhook received');
  console.log('Headers: ' + JSON.stringify(req.headers));
  console.log('Body: ' + JSON.stringify(req.body));
  console.log('HmacDigest: ' + hmacDigest);
  console.log('Signature: ' + signature);
  res.send();
});

app.listen(3000, function() {
  console.log('Server running on port 3000.');
});
```

Example Headers and Payloads

It is important to understand the payload being received. Each event contains special headers that help describe the event.

The following headers are of special importance:

Header	Description
X-Nexus-Webhook-ID	This is the event type. For example, iq:policyManagement.
X-Nexus-Webhook-Delivery	A unique UUID identifying the event.
X-Nexus-Webhook-Signature	The HMAC digest of the payload body, if an optional secret key has been configured.
X-Nexus-Webhook-Signature-Algorithm	The algorithm that calculates the HMAC digest of the body, currently only HmacSHA1.

Example Header

```
Content-Type: application/json; charset=UTF-8
User-Agent: Sonatype_CLM_Server/1.24.0-SNAPSHOT (Java 1.7.0_25; Mac OS X 10.11.5)
X-Nexus-Webhook-Signature: 687f3719b87232cf1c11b3ef7ea10c49218b6df1
X-Nexus-Webhook-Id: iq:policyManagement
X-Nexus-Webhook-Delivery: 7f4a6dde-5c68-4999-bcc0-a62f3fb8ae48
```

A payload is returned with each event type. An example application evaluation payload is shown below:

Example Payload

```
{
    'applicationEvaluation': {
        'policyEvaluationId': 'debceb1d-9209-485d-8d07-bd5390de7ef5',
        'stage': 'build',
        'ownerId': '6a454175-f55d-4d33-ba44-90ac3af2e8b8',
        'evaluationDate': '2015-05-05T23:40:12Z',
        'affectedComponentCount': 10,
        'criticalComponentCount': 2,
        'severeComponentCount': 5,
        'moderateComponentCount': 3,
        'outcome': 'fail'
    }
}
```

Event Fields

The data structure of the event payload differs by event. Event types share the following common fields:

Field	Description
Timestamp	An ISO 8601 representation of the time.
Initiator	userId or "anonymous", "system" for system events.

Policy Management Event

Policy Management events include updates to owners, policies, tags, labels, license threat groups, and owner membership mappings.

Policy Management events have the following fields:

- *action*: i.e. CREATED, UPDATED, DELETED.
- *type*: the type of entity which was updated i.e. APPLICATION, ORGANIZATION, APPLICATION_CATEGORY, LABEL, LICENSE_THREAT_GROUP, ACCESS, POLICY.
- *id*: system ID used to identify the entity which was updated.

Example Policy Management Event Payload

```
{
    'owner': {
        'id': '6a454175-f55d-4d33-ba44-90ac3af2e8b8',
        'publicId': 'webhooks_application',
        'name': 'Webhooks Application',
        'parentOwnerId': 'abaed4e0-d31e-4a67-9f71-1a8861641077',
        'type': 'APPLICATION',
        'tags': [{
            'id': '35304aee-c52f-4f66-9f7c-718e465a0e41',
            'name': 'Tag Foo',
            'description': 'A tag description.',
            'color': 'dark_red'
        }],
        'labels': [{
            'id': '35304aee-c52f-4f66-9f7c-718e465a0e41',
            'name': 'Label Foo',
            'description': 'A label description.',
            'color': 'dark_red'
        }],
        'licenseThreatGroups': [{
            'id': '35304aee-c52f-4f66-9f7c-718e465a0e41',
            'name': 'LTG Foo',
            'threatLevel': 5
        }],
        'policies': [{
            'id': '35304aee-c52f-4f66-9f7c-718e465a0e41',
            'name': 'Policy Foo',
            'threatLevel': 5
        }],
```

```
        'access': [{
            'id': '35304aee-c52f-4f66-9f7c-718e465a0e41',
            'name': 'Developers',
            'members': [{
                'type': 'USER',
                'name': 'jyoung'
            }]
        }]
    }
}
```

Application Evaluation Event

Application Evaluation events are those occurring during the lifecycle of a policy evaluation. Evaluation completed is the only evaluation event currently available.

Application Evaluation events have the following fields:

- *id*: ID of the policy evaluation.

Example Application Evaluation Event Payload

```
{
    'applicationEvaluation': {
        'policyEvaluationId': 'debceb1d-9209-485d-8d07-bd5390de7ef5',
        'stage': 'build',
        'ownerId': '6a454175-f55d-4d33-ba44-90ac3af2e8b8',
        'evaluationDate': '2015-05-05T23:40:12Z',
        'affectedComponentCount': 10,
        'criticalComponentCount': 2,
        'severeComponentCount': 5,
        'moderateComponentCount': 3,
        'outcome': 'fail'
    }
}
```

Security Vulnerability Override Management Event

Security Vulnerability Override Management events are issued when a security vulnerability override is created, updated, or deleted.

Security Vulnerability Override Management events have the following fields:

- *id*: ID of the security vulnerability override.
- *action*: CREATED, UPDATED, DELETED.

Example Security Vulnerability Override Management Event Payload

```
{
    'securityVulnerabilityOverride': {
        'id': 'd08a4954c2f942e6bbd95517030ebcf7',
        'ownerId': '6a454175-f55d-4d33-ba44-90ac3af2e8b8',
        'hash': '46c81da3225f991faa2b',
        'source': 'cve',
        'referenceId': 'CVE-2016-0788',
        'status': 'ACKNOWLEDGED',
        'comment': 'Ack'
    }
}
```

License Override Management Event

License Override Management events are issued when a license override is created, updated, or deleted.

License Override Management events have the following fields:

- *id*: ID of the license override.
- *action*: CREATED, UPDATED, DELETED.

Example License Override Management Event Payload

```
{
    'licenseOverride': {
        'id': 'cafdf38d458d461583ec6cd509dc8c31',
        'ownerId': '6a454175-f55d-4d33-ba44-90ac3af2e8b8',
        'status': 'OVERRIDEN',
        'comment': '',
        'licenseIds': [
            'Apache-2.0'
        ],
        'componentIdentifier': {
            'format': 'maven',
            'coordinates': {
                'artifactId': 'foo',
                'classifier': '',
                'extension': 'jar',
                'groupId': 'net.java.bar',
                'version': '1.9'
            }
        }
    }
}
```

Webhooks Example: IQ and Slack Integration

Using all of the information described in this topic, we wrote an article in the Sonatype Learning[121] space showing you how to build a webhook and deploy it to a Serverless[122] framework like AWS Lambda[123]. The Serverless function will consume an IQ policy evaluation event and push a message about it to Slack[124].

Check out the example here: Using Webhooks: IQ & Slack Integration[125].

sitemap.xml

template	sitemap.xml.vm

121 https://help.sonatype.com/display/SL/Sonatype+Learning
122 https://serverless.com/
123 https://aws.amazon.com/lambda/
124 https://get.slack.help/hc/en-us/articles/115004071768-What-is-Slack-
125 https://guides.sonatype.com/manuals/using-webhooks/