revenera™

OSS Inspector Plugin 1.0.0 for IntelliJ IDEA

User Guide

Legal Information

Book Name:	OSS Inspector Plugin 1.0.0 for IntelliJ IDEA User Guide
Part Number:	InspectIJ-100-IIJ00
Product Release Date:	September 2024

Copyright Notice

Intellectual Property

Restricted Rights Legend

Contents

1

Revenera OSS Inspector Plugin 1.0.0 for IntelliJ IDEA User Guide

The Revenera OSS Inspector plugin enables software developers using IntelliJ IDEA (an Integrated Development Environment, or IDE) to examine—within the IDE itself—the licenses and security vulnerabilities associated with the open-source software (OSS) components used in the product code.

This User Guide helps you get started with and use this plugin within your IntelliJ IDEA build environment.

Table 1-1 ▪ User Guide Navigation Table

Topic	Content
Installing the OSS Inspector Plugin	Requirements and instructions for installing the OSS Inspector plugin in your IntelliJ IDEA instance.
Using the OSS Inspector Plugin	Instructions for using the OSS Inspector plugin to run inspections of the OSS components in an IntelliJ IDEA build projects and explore the results.
Known Issues	Describes the current known issues in the Revenera OSS Inspector plugin available for an IntelliJ IDEA build environment.

Product Support Resources

The following resources are available to assist you:

- Revenera Product Documentation
- Revenera Community
- Revenera Learning Center
- Revenera Support

Revenera Product Documentation

You can find documentation for all Revenera products on the Revenera Product Documentation site:

https://docs.revenera.com

Revenera Community

On the Revenera Community site, you can quickly find answers to your questions by searching content from other customers, product experts, and thought leaders. You can also post questions on discussion forums for experts to answer. For each of Revenera's product solutions, you can access forums, blog posts, and knowledge base articles.

https://community.revenera.com

Revenera Learning Center

The Revenera Learning Center offers free, self-guided, online videos to help you quickly get the most out of your Revenera products. You can find a complete list of these training videos in the Learning Center.

https://learning.revenera.com

Revenera Support

For customers who have purchased a maintenance contract for their product(s), you can submit a support case or check the status of an existing case by first logging into the Revenera Community and then making selections on the **Get Support** menu, including **Open New Case** and other options.

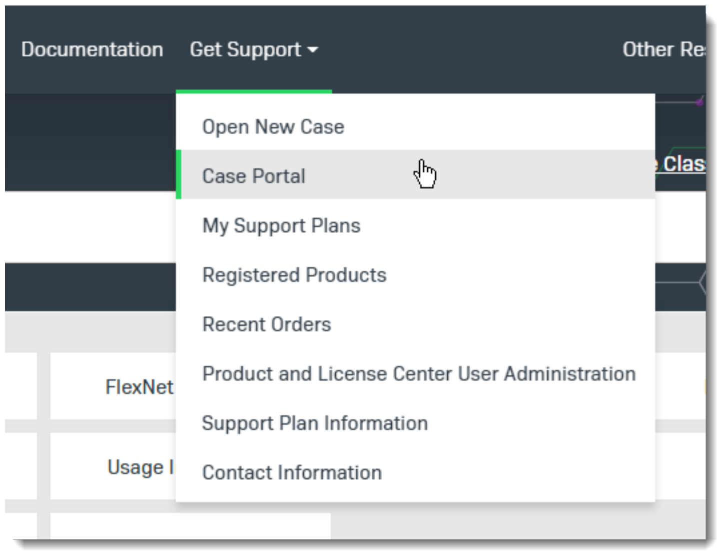


Figure 1-1: Get Support Menu of Revenera Community

Contact Us

Revenera is headquartered in Itasca, Illinois, and has offices worldwide. To contact us or to learn more about our products, visit our website at:

http://www.revenera.com

You can also follow us on social media:

- Twitter
- Facebook
- LinkedIn
- YouTube
- Instagram

2

Installing the OSS Inspector Plugin

The following information and instructions guide you through the process of installing the Revenera OSS Inspector plugin for use in your IntelliJ IDEA build environment.

Prerequisites

The following are prerequisites for using the OSS Inspector plugin.

General Prerequisites

The OSS Inspector plugin requires the following:

- IntelliJ IDEA 2021.2.4 to 2023.3.6 (Community edition).
- Supported platforms that are detailed in the below table:

Platform	Details
Windows	10 and 11 (64-bit version).
macOS	14 Sonoma with either 3.2 GHz Intel 8th-generation Core i7 or Apple Silicon M2 Pro processors.

Note ▪ *The OSS Inspector plugin is supported only for the Community edition.*

- Java Runtime Environment (JRE) 11 or later.

  Important* ▪ *This JRE should already be bundled with IntelliJ IDEA, thereby requiring no separate installation.

- A valid refresh token generated from the SBOM Management tool. For more information about obtaining this token, see Providing a Refresh Token for Authentication.
- Access to the following external host URLs required by OSS Inspector:

```
https://api.flexera.com
https://login.flexera.com/oidc/token
```

Build Module Prerequisites for OSS Inspections

The OSS Inspector plugin currently supports OSS inspections on only Gradle build modules in IntelliJ IDEA projects, along with these additional prerequisites:

- The project must be based on the Java or Kotlin language.
- The build module must contain either a `build.gradle` or `build.gradle.kts` file.

Installing the Plugin in the IDE

IntelliJ IDEA provides the following two methods for installing the OSS Inspector plugin within the IDE:

- Downloading the Plugin from Marketplace
- Manually Installing the Plugin from Your Machine

Downloading the Plugin from Marketplace

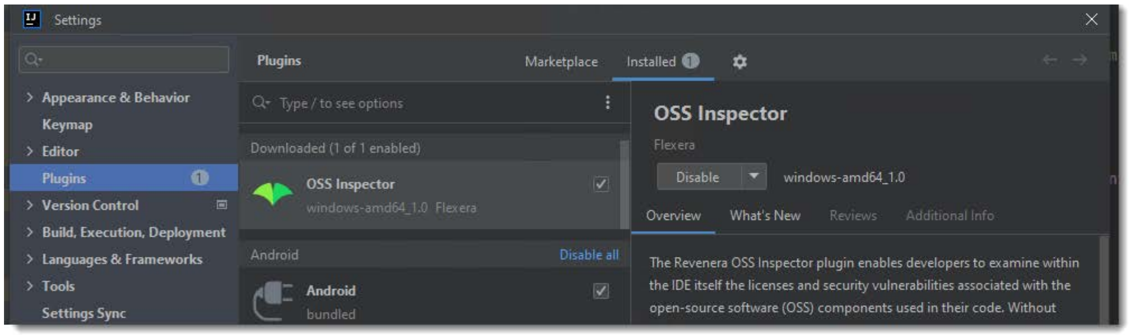
The OSS Inspector plugin for IntelliJ IDEA is currently not available for installation from the JetBrains Marketplace.

Manually Installing the Plugin from Your Machine

This process involves downloading the distribution ZIP file for the OSS Inspector plugin from Revenera and installing it in IntelliJ IDEA.

Task ***To install the plugin manually:***

1. Launch your IntelliJ IDEA application.
2. Within the IDE, press Ctrl + Alt + s to open the IDE settings, and select **Plugins**.
3. On the **Plugins** page, click ⚙ in the header, and then select **Install Plugin from Disk**.
4. Search for and select the OSS Inspector archive file that you downloaded to your machine from Revenera, and then click **Open** (or **OK**) to install it in the IDE.



The Revenera OSS Inspector plugin enables developers to examine within the IDE itself the licenses and security vulnerabilities associated with the open-source software (OSS) components used in their code. Without

Once the plugin is installed, it is immediately started (enabled). The **OSS Inspector** tab is displayed in the toolbar area at the bottom of the IDE window.

5. (Recommended) Restart the IDE to ensure that all the OSS Inspector services and components are running.

Note ▪ *Once the plugin starts up at installation, it continues to run as long as the IDE is running. Whenever the IDE is restarted, the plugin is automatically restarted.*

Disabling or Removing the OSS Inspector Plugin

Use the IntelliJ IDEA help system for instructions to disable the OSS Inspector plugin or to remove it altogether (uninstall) from the IDE.

When you disable the OSS Inspector plugin in IntelliJ IDEA, the plugin remains installed but no longer starts up automatically whenever IntelliJ is started or restarted. You can then re-enable the plugin manually as needed (by following the IntelliJ IDEA instructions). When you re-enable the plugin and then restart the IDE (per the instructions), the refresh token most recently applied to the plugin before its disablement remains in effect. However, best practice is to perform a connection test to ensure the token is still valid once you have restarted the IDE. See Re-Validating the Current Refresh TokenRe-Validating the Current Refresh Token.

Should you uninstall the OSS Inspector plugin from IntelliJ IDEA and later re-install it in the IDE, you must re-enter the refresh token (or obtain a new one) in order to use the plugin. See Providing a Refresh Token for Authentication for instructions.

Important ▪ *In general, you should always restart the IDE after you install/uninstall or enable/disable the OSS Inspector plugin, whether or not this step is listed in the instructions.*

Providing a Refresh Token for Authentication

The configuration process for the OSS Inspector plugin involves specifying a valid refresh token used to authenticate the user who is running the plugin.

- About the Refresh Token
- Obtaining the Refresh Token
- Specifying the Refresh Token for the Plugin

- Re-Validating the Current Refresh Token

About the Refresh Token

The **refresh token** is a long-lived credential that enables the OSS Inspector plugin to access a Revenera internal service that retrieves metadata and security vulnerability information associated with the OSS dependencies used in a project. The token is entered once for the plugin and persists throughout all InelliJ IDEA sessions. If you close the IDE and then reopen it, the token still persists.

***Note** ▪ Whenever you restart the IDE, best practice is to run a connection test to ensure the token is still valid (and provide a new one if necessary). In this way, you can initiate an OSS inspection without the possibility of being interrupted because of an invalid token. See Re-Validating the Current Refresh Token.*

Important Information About Token Usage

Note the following about the usage of a refresh token:

- For security reasons, the refresh token expires in one year if it is not used within that one year.
- The token will never expire if used at least once per year.
- There is no limit to the number of active refresh tokens that a user can have.

Obtaining the Refresh Token

To perform the OSS inspections, the plugin needs to be configured with a valid refresh token, which can be generated from the SBOM Management tool.

For more information, contact Revenera Support.

Specifying the Refresh Token for the Plugin

Once you obtain the refresh token needed to run the OSS Inspector plugin, you must specify this token in the IntelliJ IDEA settings.

Task ***To specify the refresh token for the plugin within IntelliJ IDEA settings:***

1. Copy the refresh token that you have securely stored. (If you have not already obtained a valid refresh token, see Obtaining the Refresh Token.)
2. Within the IDE, press Ctrl + Alt + s to open the IDE settings, and select **OSS Inspector Settings**.
3. On the **OSS Inspector Settings** page, paste the refresh token in the **Refresh Token** field. The token value is automatically masked.
4. (Recommended) Click **Test Connection** to ensure the token is valid. One of the following happens:
    - If the refresh token is valid, a message indicates that authentication was successful.

- If the token is invalid, an "Invalid refresh token!" message is displayed and the **Refresh Token** field is cleared. Try copying and reentering the token or obtain a new token, and then repeat the previous steps.
- If the test indicates the that service is down, a wait for the service to restart; or restart the IDE and repeat the steps in this procedure.

Note ▪ *Best practice is to test the connection at the time you provide the token. This will avoid an "Authentication failed" error should the plugin find the current token invalid when you attempt to initiate an inspection.*

5. Do one of the following.
   - Click **OK** to save the refresh token to the plugin and close the IDE settings.
   - Click **Apply** to save the token and keep the IDE settings open to continue IDE configuration.
   - Click **Cancel** not to save the current token and close the IDE settings.

Note ▪ *No token validation takes place when you close the* ***OSS Inspector Settings*** *page, even if you click* ***OK*** *or* ***Apply*** *to save the token. Validation takes place only when you run a* ***Test Connection*** *from this page before closing it.*

Re-Validating the Current Refresh Token

Whenever you restart IntelliJ IDEA, best practice is to re-validate the current refresh token for the OSS Inspector plugin before running OSS inspections on projects. While not required, this step re-ensures that nothing has happened with the token's validity during shutdown or disablement.

Task ***To re-validate the current refresh token:***

1. After restarting the IDE, press Ctrl + Alt + s to open the IDE settings.
2. Select **OSS Inspector Settings**.

   The current refresh token in the **Refresh Token** field on the **OSS Inspector Settings** page is masked.
3. Click **Test Connection**.
4. Perform the appropriate step:
   - If a success message is displayed, click **OK** to close the IDE settings and proceed with OSS inspections.
   - If the test indicates the that service is down, wait for the service to restart; or restart the IDE and repeat the steps in this procedure.
   - If the test indicates that the refresh token is invalid, obtain a new token and repeat the steps in Specifying the Refresh Token for the Plugin.

3 Using the OSS Inspector Plugin

The following information and instructions describe how to use the Revenera OSS Inspector plugin available for an IntelliJ IDEA build environment:

- Purpose of the Plugin
- Running an OSS Inspection on a Build Module
- Exploring Inspection Results
- Properties Displayed on OSS Inspector Tab
- OSS Reinspection for Modules and Sub-Modules

Purpose of the Plugin

The Revenera OSS Inspector plugin enables developers using IntelliJ IDEA (an Integrated Development Environment, or IDE) to examine—within the IDE itself—the licenses and security vulnerabilities associated with the open-source software (OSS) components used in their application code (represented by a project in the IDE). Developers can immediately assess security risks to determine whether they require further review and remediation without leaving the IDE.

Running an OSS Inspection on a Build Module

The following sections describe how to trigger an inspection of the OSS dependencies used in a Gradle build module in an IntelliJ IDEA project.

- Initiating an OSS Inspection
- OSS Inspector Checks Performed and Possible Issues Encountered

To ensure that the build module meets the requirements for an OSS inspection, see Build Module Prerequisites for OSS Inspections.

Initiating an OSS Inspection

Use these steps to initiate an OSS inspection of the OSS components used in a selected Gradle build module located in a project in IntelliJ IDEA.

Task ***To run an OSS inspection on a Gradle build module:***

1. (Optional) As a best practice if IntelliJ IDEA has been shut down and restarted, re-validate the current refresh token for the plugin (see Re-Validating the Current Refresh Token).

2. Open the appropriate project in the IDE.

   Note ▪ *The OSS Inspector requires that the project be Java- or Kotlin-based. Otherwise, an error is displayed (see OSS Inspector Checks Performed Before the Inspection Starts).*

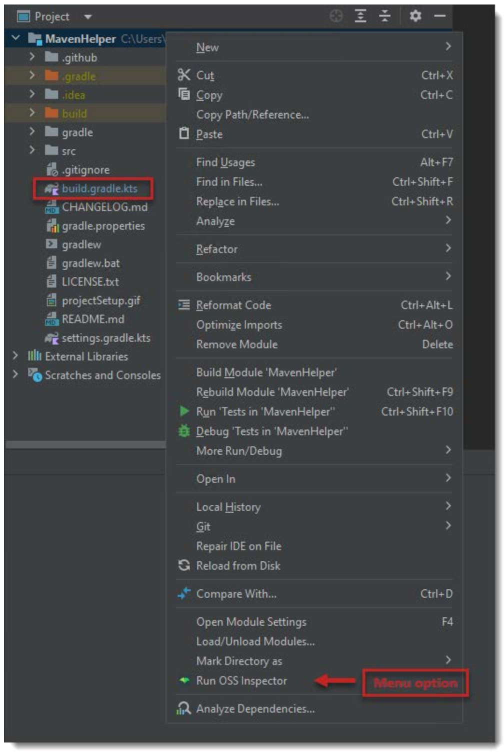
3. In the project codebase, locate the specific folder for the Gradle build module you want to inspect. This can be the folder for a single build module or a build module that encapsulates additional build modules (that is, sub-modules).

   Note ▪ *The OSS Inspector requires that the selected Gradle build module contain a `build.gradle` or `build.gradle.kts` file. Otherwise, an error is displayed (see OSS Inspector Checks Performed Before the Inspection Starts).*

4. Right-click the folder, and select **Run OSS Inspector**.

   In the following example, the build module is the top-level project folder `MavenHelper`, which contains the `build.gradle.kts` file.

   For situations when the **Run OSS Inspector** option is not available for build modules, see Availability of the “Run OSS Inspector” Option.



If authentication is successful and the project and selected build module meet certain criteria (see OSS Inspector Checks Performed Before the Inspection Starts), the message “Code inspection initiated”, along with an inspection progress bar, is displayed at the bottom of the IDE window.

Once initiated, the inspection proceeds to collect the list of dependencies used in the build module and to retrieve metadata and any security vulnerability information associated with each dependency. When the inspection process is complete, the **OSS Inspector** tab is automatically opened, enabling you to examine the inspection results. See Exploring Inspection Results for further instructions.

For issues that the inspection process might encounter, see Possible Issues Encountered During the OSS Inspection.

Availability of the “Run OSS Inspector” Option

The following describes the conditions under which the **Run OSS Inspector** option is not available for build modules that are otherwise qualified for inspections:

- Only one OSS inspection can be triggered at any one time across project instances in the IDE. If an OSS inspection is currently in progress on a given project in the IDE, the **Run OSS Inspector** option is disabled for all other build modules in the same project or in another project. Once the current inspection is complete, the option is re-enabled.
- The **Run OSS Inspector** option to run an OSS inspection on a given build module in a project is disabled if a build or indexing process is currently in progress on the project. Once the process is complete, the option is re-enabled.
- The **Run OSS Inspector** option is available only at the folder level for a build module, not at a file level.

OSS Inspector Checks Performed and Possible Issues Encountered

The following sections describes the checks that the OSS Inspector performs before a given inspection is initiated and highlights certain issues that the Inspector can encounter during the inspection process.

- OSS Inspector Checks Performed Before the Inspection Starts
- Possible Issues Encountered During the OSS Inspection

OSS Inspector Checks Performed Before the Inspection Starts

Once a user clicks **Run OSS Inspector** to start an inspection on a build module, the OSS Inspector plugin performs the following checks before it will initiate the inspection. If the project and build module passes all these checks, the "Code inspection initiated" message is displayed, along with an inspection progress bar, at the bottom of the IDE window. However, if the project or build module fails any *one* of these checks, an appropriate error message is displayed and the inspection is not initiated.

User Authentication

If the refresh token provided for the plugin for authentication is invalid or no if token has been provided, an "Authentication failed..." error message is displayed, and the inspection does not initiate. Before you can attempt another inspection, you must provide a valid refresh token, as described in Providing a Refresh Token for Authentication.

Service Status

In some cases authentication fails with the message "Core Service unavailable", meaning that the service used by the plugin is currently not running. In this case, restart IntelliJ IDEA in an attempt to restart the service, and then repeat the steps in Initiating an OSS Inspection.

Project and Module Requirements Check

The plugin runs the following checks on the project and build module:

- Checks that the selected project is Java or Kotlin project. If the project is not based on one of these languages, the message "Not a Java/Kotlin project..." is displayed.
- Checks that the selected folder is a Gradle module or submodule (that is, it contains a `build.gradle` or `build.gradle.kts` file). If the module contains neither of these files, the message "Gradle build file is not present..." is displayed.
- For macOS environment, the plugin requires that the Gradle Wrapper script (gradlew) located in the root folder of the selected project has executable permissions. This requirement is crucial for the plugin to function properly. Before proceeding with the OSS Inspection, verify that the gradlew script has been set as executable. Failing to do so may cause the OSS inspection process to fail.

When the plugin cannot identify either the project language or the build tool, the message "Could not identify project language or build tool" might be displayed.

Possible Issues Encountered During the OSS Inspection

The section highlights certain errors that the OSS Inspector can encounter during the inspection process (before the results are displayed). These issues can cause partial or no results to display.

“Could not resolve project dependencies”

This error is displayed if the OSS Inspector is unable to identify any dependencies in the build module or sub-module(s) included in the inspection.

“Could not resolve dependencies...for module <module name>”

If the OSS Inspector fails to resolve dependencies for a specific module or sub-module, it displays the message: “Could not resolve project dependencies. Failed to resolve dependencies for module: <module name>”.

However, the inspection continues for any remaining build modules included in the inspection.

“Failed to inspect code”

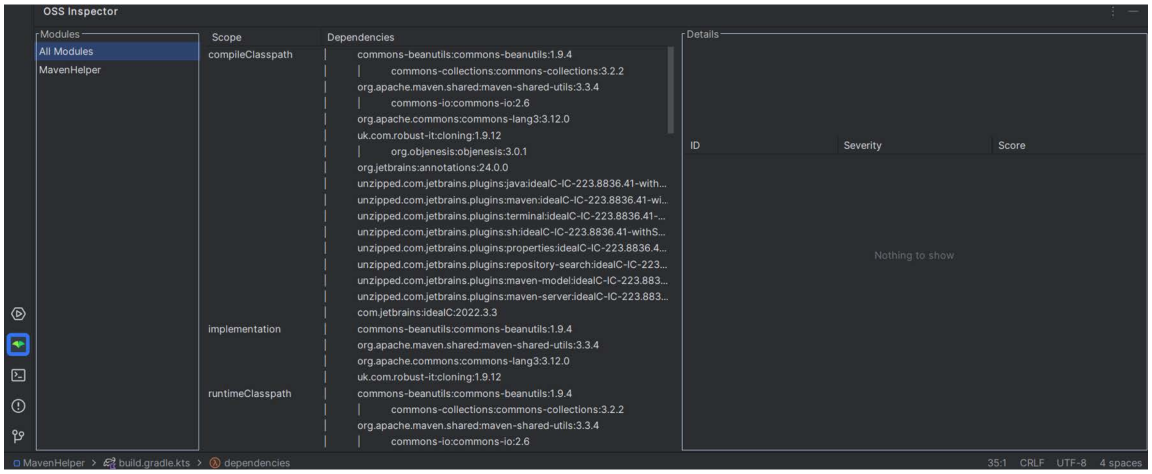
This generic message is displayed if the entire inspection fails once it is initiated.

Exploring Inspection Results

When the OSS inspection completes, the **OSS Inspector** tab opens, showing the results of the inspection. The tab includes three panes. When the tab first opens, only the **Modules** pane (on the left) is populated, showing the list of modules that were inspected. To explore the dependencies in these modules, use the following procedure.

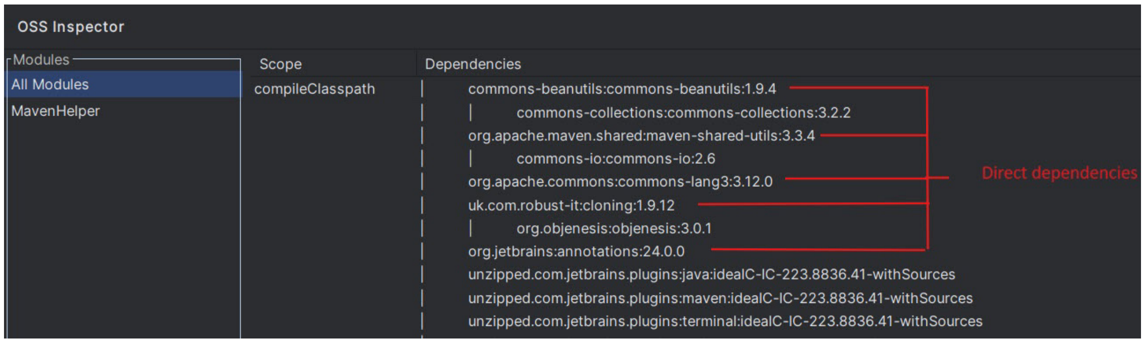
Task ***To explore the inspection results:***

1. In the **Modules** pane on the **OSS Inspector** tab, you can select a module to view its dependencies in the middle pane. You can also select the **All Modules** property to view a consolidated list of all dependencies from all modules. In this example, only one module was inspected.



```
Scope               Dependencies
compileClasspath    | commons-beanutils:commons-beanutils:1.9.4
                    | |    commons-collections:commons-collections:3.2.2
                    | org.apache.maven.shared:maven-shared-utils:3.3.4
                    | |    commons-io:commons-io:2.6
                    | org.apache.commons:commons-lang3:3.12.0
                    | uk.com.robust-it:cloning:1.9.12
                    | |    org.objenesis:objenesis:3.0.1
                    | org.jetbrains:annotations:24.0.0
                    | unzipped.com.jetbrains.plugins:java:idealC-IC-223.8836.41-with...
                    | unzipped.com.jetbrains.plugins:maven:idealC-IC-223.8836.41-wi...
                    | unzipped.com.jetbrains.plugins:terminal:idealC-IC-223.8836.41-...
                    | unzipped.com.jetbrains.plugins:sh:idealC-IC-223.8836.41-withS...
                    | unzipped.com.jetbrains.plugins:properties:idealC-IC-223.8836.4...
                    | unzipped.com.jetbrains.plugins:repository-search:idealC-IC-223...
                    | unzipped.com.jetbrains.plugins:maven-model:idealC-IC-223.883...
                    | unzipped.com.jetbrains.plugins:maven-server:idealC-IC-223.883...
                    | com.jetbrains:idealC:2022.3.3
implementation      | commons-beanutils:commons-beanutils:1.9.4
                    | org.apache.maven.shared:maven-shared-utils:3.3.4
                    | org.apache.commons:commons-lang3:3.12.0
                    | uk.com.robust-it:cloning:1.9.12
runtimeClasspath    | commons-beanutils:commons-beanutils:1.9.4
                    | |    commons-collections:commons-collections:3.2.2
                    | org.apache.maven.shared:maven-shared-utils:3.3.4
                    | |    commons-io:commons-io:2.6
```

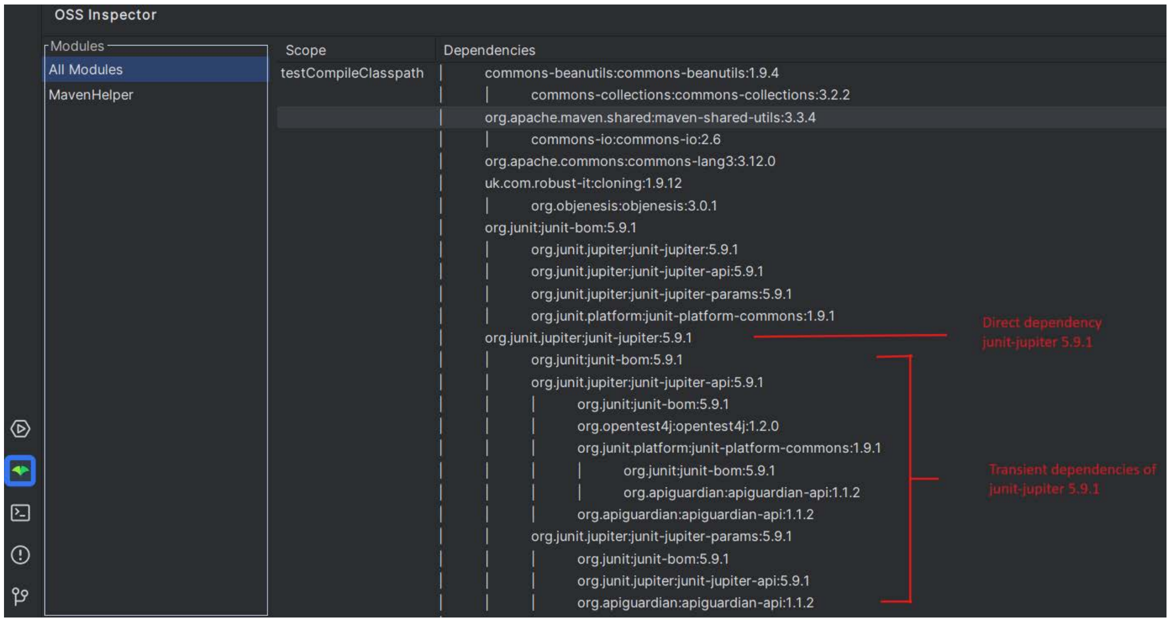
The dependencies are grouped according to scope type. For a given scope, the dependencies are shown in a tree format, depicting *direct dependencies* at the first level.



```
compileClasspath  commons-beanutils:commons-beanutils:1.9.4
                  |    commons-collections:commons-collections:3.2.2
                  org.apache.maven.shared:maven-shared-utils:3.3.4
                  |    commons-io:commons-io:2.6
                  org.apache.commons:commons-lang3:3.12.0
                  uk.com.robust-it:cloning:1.9.12
                  |    org.objenesis:objenesis:3.0.1
                  org.jetbrains:annotations:24.0.0
                  unzipped.com.jetbrains.plugins:java:idealC-IC-223.8836.41-withSources
                  unzipped.com.jetbrains.plugins:maven:idealC-IC-223.8836.41-withSources
                  unzipped.com.jetbrains.plugins:terminal:idealC-IC-223.8836.41-withSources
```

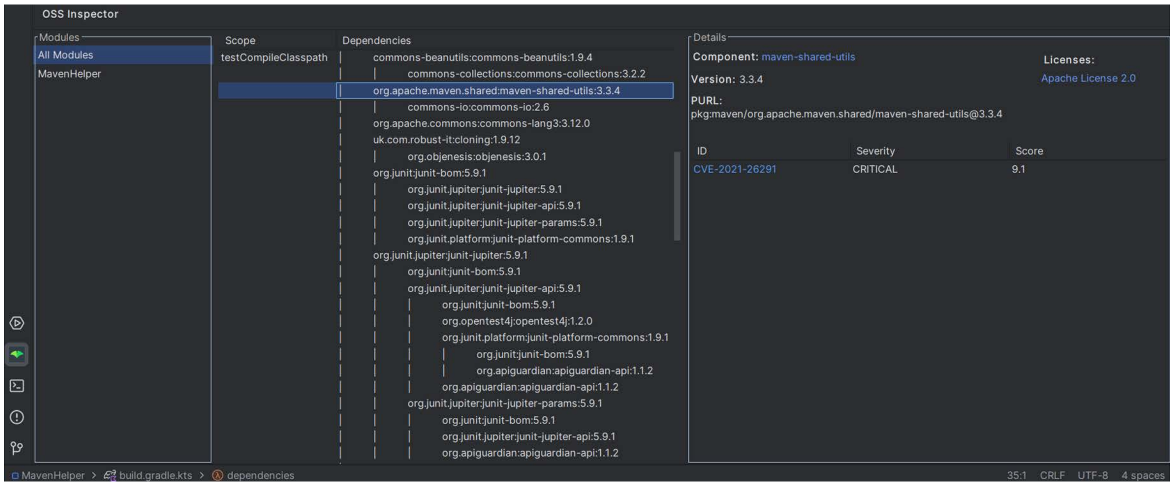
Each direct dependency represents an OSS component directly used by the main OSS component (represented by the project itself) that is integrated in your application.

Beneath a given direct dependency, the list of *transient dependencies* (if any) is displayed. Transient dependencies are those OSS components used by the direct dependency. They are listed in levels to reflect their association with each other and ultimately their association with the direct dependency.


OSS Inspector
Modules
All Modules
MavenHelper
Scope
Dependencies
Direct dependency junit-jupiter 5.9.1
Transient dependencies of junit-jupiter 5.9.1

```
testCompileClasspath  commons-beanutils:commons-beanutils:1.9.4
                      |    commons-collections:commons-collections:3.2.2
                      org.apache.maven.shared:maven-shared-utils:3.3.4
                      |    commons-io:commons-io:2.6
                      org.apache.commons:commons-lang3:3.12.0
                      uk.com.robust-it:cloning:1.9.12
                      |    org.objenesis:objenesis:3.0.1
                      org.junit:junit-bom:5.9.1
                      |    org.junit.jupiter:junit-jupiter:5.9.1
                      |    org.junit.jupiter:junit-jupiter-api:5.9.1
                      |    org.junit.jupiter:junit-jupiter-params:5.9.1
                      |    org.junit.platform:junit-platform-commons:1.9.1
                      org.junit.jupiter:junit-jupiter:5.9.1
                      |    org.junit:junit-bom:5.9.1
                      |    org.junit.jupiter:junit-jupiter-api:5.9.1
                      |    |    org.junit:junit-bom:5.9.1
                      |    |    org.opentest4j:opentest4j:1.2.0
                      |    |    org.junit.platform:junit-platform-commons:1.9.1
                      |    |    |    org.junit:junit-bom:5.9.1
                      |    |    |    org.apiguardian:apiguardian-api:1.1.2
                      |    |    org.apiguardian:apiguardian-api:1.1.2
                      |    org.junit.jupiter:junit-jupiter-params:5.9.1
                      |    |    org.junit:junit-bom:5.9.1
                      |    |    org.junit.jupiter:junit-jupiter-api:5.9.1
                      |    |    org.apiguardian:apiguardian-api:1.1.2
```

2. Select an OSS dependency to view its details in the **Details** pane (on the left). The details include the dependency's component information, its license(s), and any associated security vulnerabilities.



OSS Inspector

Modules
- All Modules
- MavenHelper

```
Scope                  Dependencies
testCompileClasspath   |    commons-beanutils:commons-beanutils:1.9.4
                       |    |    commons-collections:commons-collections:3.2.2
                       |    org.apache.maven.shared:maven-shared-utils:3.3.4
                       |    |    commons-io:commons-io:2.6
                       |    org.apache.commons:commons-lang3:3.12.0
                       |    uk.com.robust-it:cloning:1.9.12
                       |    |    org.objenesis:objenesis:3.0.1
                       |    org.junit:junit-bom:5.9.1
                       |    |    org.junit.jupiter:junit-jupiter:5.9.1
                       |    |    org.junit.jupiter:junit-jupiter-api:5.9.1
                       |    |    org.junit.jupiter:junit-jupiter-params:5.9.1
                       |    |    org.junit.platform:junit-platform-commons:1.9.1
                       |    org.junit.jupiter:junit-jupiter:5.9.1
                       |    |    org.junit:junit-bom:5.9.1
                       |    |    org.junit.jupiter:junit-jupiter-api:5.9.1
                       |    |    |    org.junit:junit-bom:5.9.1
                       |    |    |    org.opentest4j:opentest4j:1.2.0
                       |    |    |    org.junit.platform:junit-platform-commons:1.9.1
                       |    |    |    |    org.junit:junit-bom:5.9.1
                       |    |    |    |    org.apiguardian:apiguardian-api:1.1.2
                       |    |    |    org.apiguardian:apiguardian-api:1.1.2
                       |    |    org.junit.jupiter:junit-jupiter-params:5.9.1
                       |    |    |    org.junit:junit-bom:5.9.1
                       |    |    |    org.junit.jupiter:junit-jupiter-api:5.9.1
                       |    |    |    org.apiguardian:apiguardian-api:1.1.2
```

Details

Component: maven-shared-utils

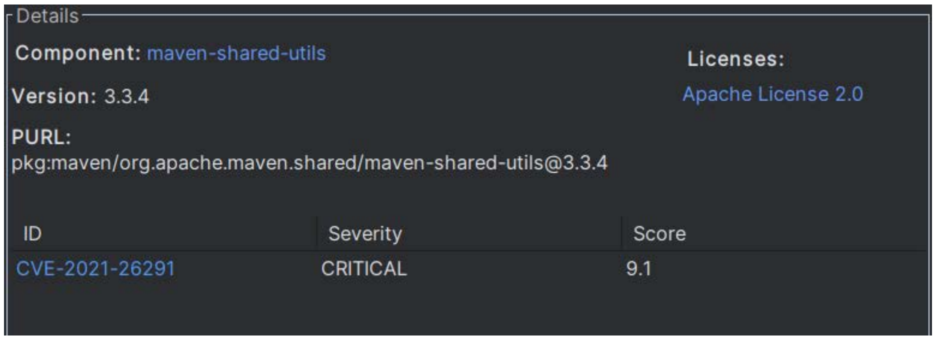
Version: 3.3.4

PURL:
pkg:maven/org.apache.maven.shared/maven-shared-utils@3.3.4

Licenses:
Apache License 2.0

ID	Severity	Score
CVE-2021-26291	CRITICAL	9.1

3. In the dependency details, click the hyperlinked component name, license, or vulnerability ID to open your browser to that entity's external website for further information.



Details

Component: maven-shared-utils

Version: 3.3.4

PURL:
pkg:maven/org.apache.maven.shared/maven-shared-utils@3.3.4

Licenses:
Apache License 2.0

ID	Severity	Score
CVE-2021-26291	CRITICAL	9.1

Properties Displayed on OSS Inspector Tab

The following are the details used to describe OSS dependencies on the **OSS Inspector** tab.

Table 3-1 ▪ Dependency Details on the OSS Inspector Tab

Pane	Property	Description
Modules (left pane)	This pane lists all Gradle build modules that were inspected for OSS dependencies. When you select a module or **All Modules** property from the list, its related dependencies are listed on the **Dependencies** pane.	
	All Modules	Selecting this property enables you to view a consolidated list of all dependencies on the **Dependencies** pane—pertaining to all Gradle build modules inspected for OSS dependencies.

Table 3-1 ▪ Dependency Details on the OSS Inspector Tab (cont.)

Pane	Property	Description
Dependencies (middle pane)	This pane lists the OSS dependencies used in the build module. The dependencies are grouped by the scope.	
	Scope	The "when and where" that the dependency is used by the application (for example, during compilation only, runtime only, compilation and runtime, testing, and more). A dependency can be used in one or more scopes (and therefore can be listed multiple times in this pane for a given build module).
	Dependencies	The name (component and version) of each dependency. The dependencies are organized by the scope in which they are used. For each scope, the dependencies are shown in a tree format: ● Direct dependencies are depicted at the first level in the tree. Each direct dependency represents an OSS component directly used by the main OSS component (represented by the project itself) that is integrated in your application. ● Under a given direct dependency is its list of transient dependencies (if any). Transient dependencies are those OSS components used by the direct dependency. ● The transient dependencies for a direct dependency are listed in one or more levels to reflect their association with each other and ultimately with the direct dependency. When you select a dependency listed for a specific scope, information about that dependency is displayed in the **Details** pane.

Table 3-1 ▪ Dependency Details on the OSS Inspector Tab (cont.)

Pane	Property	Description
Details (right pane)	The following information is shown the OSS dependency selected in the **Dependencies** pane.	
	Component	The component name of the OSS dependency. Click the hyperlinked component name to open your browser to the web page of the component's third-party project or repository within the appropriate forge.
	Version	The component version of the OSS dependency.
	PURL	The PURL (package URL) for the component. A PURL is an attempt to standardize existing methods (syntax, naming conventions, and other formats) to reliably identify and locate software packages. In other words, the PURL tries to identify and locate a software package using the most universal and uniform format possible—suitable across programing languages, package managers, packaging conventions, tools, APIs, and databases. Refer to the package-url/purl-spec page in GitHub for additional information.
	Licenses	The license(s) associated with the component version. (If available, the SPDX short name is shown for each license.) Click the hyperlinked license name to view detailed information about the license in the Linux Foundation Projects SPDX license database.
	Vulnerability ID	The ID of the security vulnerability in the format of the advisory organization that reported it: ● For a vulnerability reported by the NVD, the ID uses the CVE (Common Vulnerabilities and Exposures) format. ● For a vulnerability reported by another research organization, the ID uses the format specific to that organization. You can click the hyperlinked URL to open the website for the source advisory to explore more the information about the vulnerability and its possible fixes.

Table 3-1 ▪ Dependency Details on the OSS Inspector Tab (cont.)

Pane	Property	Description
	Severity	The severity level of a specific security vulnerability is pulled from the National Vulnerability Database (NVD) or from another advisory database used to identify the vulnerability. The severity is based on the vulnerability's CVSS (Common Vulnerability Scoring System) score. ***Note ▪*** *OSS Inspector uses the CVSS v3.x scoring system, which includes v3.1 and v3.0. A given security vulnerability can have either a 3.1 or 3.0 score, not both.* • **CRITICAL**—CVSS v3.*x* score 9.0 - 10.0 • **HIGH**—CVSS v3.*x* score7.0 - 8.9 • **MEDIUM**—CVSS v3.*x* score 4.0 - 6.9 • **LOW**—CVSS v3.*x* score 0.1 - 3.9 • **N/A**—No severity available due to lack of a CVSS v3.*x* score
	Score	The vulnerability's CVSS (Common Vulnerability Scoring System) score.

OSS Reinspection for Modules and Sub-Modules

The OSS reinspection for the modules and sub-modules of a project involves re-analyzing the open-source software components to identify potential vulnerabilities, ensure compliance with licensing requirements, and maintain the integrity of the software at both the module and sub-module levels.

Performing an OSS reinspection on a root module triggers a rescan for that module (including its sub-modules) only for the following conditions:

- When the `libs.version.toml` file is updated.
- When the `build.gradle` or `build.gradle.kts` files are updated.

However, performing an OSS reinspection on a sub-module always triggers a rescan for that sub-module, regardless of any changes in the top-level `libs.versions.toml`, sub-module's `build.gradle`, or sub-module's `build.gradle.kts` files.

Note ▪ *Consider the following information pertaining to the `libs.version.toml` file during the OSS reinspection process:*

- *The `libs.versions.toml` file must be present within the Gradle folder of the project, either at the `/gradle/libs.versions.toml` directory or in the root folder.*

- *Only one `libs.versions.toml` file will be considered throughout the entire project. Any additional .toml files found at the module or sub-module level (such as, `test-libs.versions.toml`, `abc.versions.toml`) will be ignored.*

4

Known Issues

The following information describes the current known issues in the Revenera OSS Inspector plugin available for an IntelliJ IDEA build environment:

STRATUS-1425: Rescan triggers for a top-level module without libs.version.toml file

A top-level module and all its sub-modules are being rescanned, even if the `libs.version.toml` file is missing and there have been no changes to the `build.gradle` or `build.gradle.kts` files.

Workaround: None exists.

STRATUS-1426: Duplicate modules are displayed in the “Modules” pane

Upon completion of an OSS inspection process, duplicate modules are displayed in the **Modules** pane of the **OSS Inspector** tab.

Workaround: None exists.

STRATUS-1427: Failure to resolve dependencies during an initial scan

All dependencies for a module are failed to resolved in the initial scan; they are only resolved after multiple rescans.

Workaround: None exists.

STRATUS-1428: Discrepancy in dependency version between “OSS Inspector” Tab and Gradle resolution

A discrepancy has been observed between the version of a dependency displayed in the **Dependencies** pane of the **OSS Inspector** tab and the version resolved through Gradle.

Workaround: None exists.

STRATUS-1429: Port conflict issue with Zaga server in multiple versions of IntelliJ IDEA

Multiple versions of IntelliJ IDEA on macOS operating system fail to use different ports for the Zaga server (a server used internally by the OSS Inspector plugin for its operation). Instead they attempt to use the same port, leading to a conflict.

Workaround: None exists.