

Parasoft Findings for SonarQube

In this section:

- Introduction
- Requirements
- Installing the Parasoft Findings Plugin
- Setting the Root Path to the Parasoft Product
- Activating Parasoft Profiles
- Setting the Report Path
- Uploading Project Results
- Viewing Test Execution Results
- Unit Test Execution and Coverage

View online: <https://docs.parasoft.com/x/z7K9Bw>

Introduction

The Parasoft Findings Plugin for SonarQube allows you to view static analysis test results within SonarQube. It grants SonarQube the ability to analyze data from Parasoft XML reports and use it to report bugs, vulnerabilities, or code smells from within SonarQube. See [Uploading Project Results](#) for more details.

The plugin can consume the following report types:

- Parasoft Analyzers static analysis XML reports generated by Parasoft C/C++test, Jtest and dotTEST tools.

Requirements

- A Parasoft product must be installed on the same machine as the SonarQube server since the plugin loads rules from a Parasoft product installation.
- For C/C++test, the Standard edition must be used as the Professional edition does not include the required rule files.
- SonarQube versions 8.9-9.5 are supported.
- Language projects versions 2022.2+ are supported.

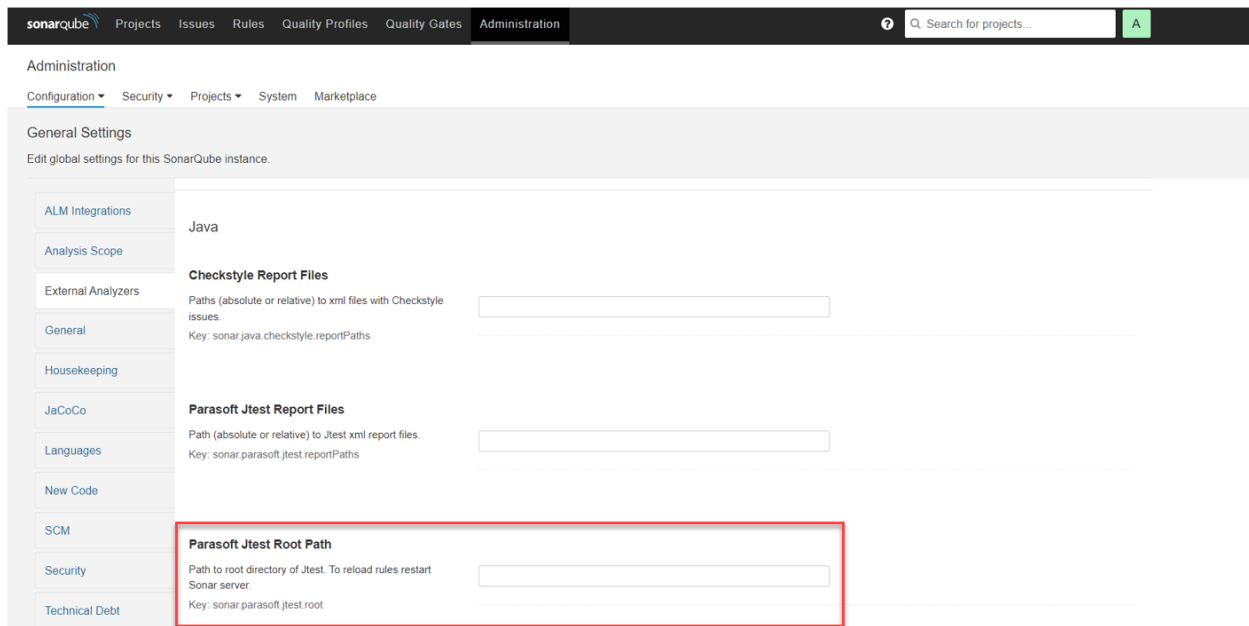
Installing the Parasoft Findings Plugin

1. Go to the `integration/sonarqube` directory of your Parasoft product and locate the Parasoft plugin jar:
 - `com.parasoft.xtest.reports.sonar-10.6.1-RELEASE.jar`
2. Copy the plugin jar into the SonarQube `extensions/plugins` directory.
3. Restart the SonarQube server.

See <https://docs.sonarqube.org/latest/setup/install-plugin/> for more details.

Setting the Root Path to the Parasoft Product

1. In the SonarQube web UI go to **Administration > Configuration > General Settings > External Analyzers**.
2. Locate the **Root Path** setting under the appropriate programming language.



3. In the field enter the absolute path to the Parasoft product installation and click **Save**.
4. Restart the SonarQube server so that the rule definitions are loaded.
5. After the server has restarted, go to **Quality Profiles** in the web UI and select the built-in Parasoft profile under the appropriate supported language. Verify that the rule definitions are loaded successfully under the profile.

If zero or only one rule is loaded, then loading of rules failed. In this case:

- a. Check that the root path is correct.
- b. Review the SonarQube web server logs for any error messages.

Activating Parasoft Profiles

There are two ways to enable the Parasoft profile for your projects:

- To set a Parasoft profile as the default for new projects, go to **Quality Profiles** and select **Set as Default** from the gear menu to the right of the profile.
- To set a Parasoft profile for a given project, go to **Quality Profiles** section in the project settings. Select the **Change Profile** button on the right side of the appropriate language.

Setting the Report Path

For Parasoft results to be uploaded, the path to the report files must be set. There are several ways to configure this.

- To configure the path globally go to **Administration > Configuration > General Settings > External Analyzers** and find the **Report Files** setting for your Parasoft product.

The screenshot shows the SonarQube Administration interface. The top navigation bar includes 'sonarqube', 'Projects', 'Issues', 'Rules', 'Quality Profiles', 'Quality Gates', and 'Administration'. The 'Administration' section is expanded, showing 'Configuration', 'Security', 'Projects', 'System', and 'Marketplace'. The 'Configuration' section is further expanded to show 'General Settings'. The 'General Settings' page has a sidebar with various categories: 'ALM Integrations', 'Analysis Scope', 'External Analyzers', 'General', 'Housekeeping', 'JaCoCo', 'Languages', 'New Code', 'SCM', 'Security', and 'Technical Debt'. The 'External Analyzers' category is selected. The main content area shows settings for 'Java'. Under 'Checkstyle Report Files', there is a text input field for the path and a key 'sonar.java.checkstyle.reportPaths'. Below this, the 'Parasoft Jtest Report Files' section is highlighted with a red box. It contains a text input field for the path and a key 'sonar.parasoft.jtest.reportPaths'. At the bottom, the 'Parasoft Jtest Root Path' section has a text input field for the path and a key 'sonar.parasoft.jtest.root'.

Enter the path to the report.xml file of your project. For example, for Jtest using Maven:

```
target/jtest/report.xml
```

You can add multiple paths, each in a separate field. When the Parasoft scanner runs, results will be loaded from each valid report file path.

- To configure the path for a project, go to the **External Analyzers** section in the project's settings.
- To configure the path when running the sonar scanner, include the report path's settings key. For example, for Jtest using Maven:


```
mvn sonar:sonar -Dsonar.parasoft.jtest.reportPaths=target/report.xml...<additional sonar settings>
```

You can include multiple report files:

```
mvn sonar:sonar -Dsonar.parasoft.jtest.reportPaths=target/report.xml  
-Dsonar.parasoft.jtest.reportPaths=target/report_2.xml...<additional sonar settings>
```

Uploading Project Results

To upload Parasoft results for a project, first run the command which generates the report.xml file, and then run the usual Sonar scanner command. For example, with a Maven project setup using Jtest, a single command can be run from the root directory of your project:

```
mvn jtest:jtest sonar:sonar \
-Dsonar.projectKey=project key> \
-Dsonar.host.url=<server url> \
-Dsonar.login=<login token>
```

See the documentation for your Parasoft product for details on how to run static analysis and generate a report:

- For Jtest: <https://docs.parasoft.com/display/JTEST20221/Running+Static+Analysis+1>
- For dotTEST: <https://docs.parasoft.com/display/CPPTTEST20221/Running+Static+Analysis+1>
- For C/C++test: <https://docs.parasoft.com/display/DOTTEST20221/Running+Static+Analysis+1>

See <https://docs.sonarqube.org/latest/analysis/overview/> for details on how to run the Sonar scanner on projects using different build systems.

Viewing Test Execution Results

After the results are uploaded, they can be viewed in the project in the **Issues** tab.

The screenshot displays the SonarQube web interface for a project named "Demo Project". The top navigation bar includes links for Projects, Issues, Rules, Quality Profiles, Quality Gates, and Administration. A search bar is present on the right. The main header shows the project name, a star icon, and a branch "master". A yellow badge indicates "Last analysis had 1 warning" on August 8, 2022, at 10:16 AM, with version 1.0.0. The left sidebar contains a "Filters" section with a "Clear All Filters" button. Under "Type", "BUG" is selected, showing 128 items. Under "Severity", "Blocker" (70), "Critical" (22), and "Major" (36) are listed. The main content area shows a list of issues. The first section, titled "src/.../java/examples/flowanalysis/AlwaysCloseGSS.java", contains four issues: two "Security context not disposed: GSSManager.getInstance().createContext(tokens)" and two "'context' may possibly be null". The second section, titled "src/.../java/examples/flowanalysis/AlwaysCloseImages.java", contains three issues: two "Image resource not closed: imgReader" and one "'imgReader' may possibly be null". Each issue entry includes a checkbox, a bug icon, a severity level (Blocker), a status (Open), an assigned user (Not assigned), a comment link, a timestamp (4 minutes ago), a line number (L13, L29, L16, L31), and a link to the issue details.

Filters [Clear All Filters](#)

Type **BUG** [Clear](#)

Severity

- Blocker 70
- Critical 22
- Major 36
- Minor 0
- Info 0

Scope

Resolution

Status

Security Category

Creation Date

Language

Issues [All](#)

[Bulk Change](#)

[1 / 128 issues](#) [0 effort](#)

src/.../java/examples/flowanalysis/AlwaysCloseGSS.java

- ☐ **Security context not disposed: GSSManager.getInstance().createContext(tokens)** Why is this an issue? 4 minutes ago L13 [bd.res, parasoft](#)
- ☐ **Security context not disposed: GSSManager.getInstance().createContext(tokens)** Why is this an issue? 4 minutes ago L13 [bd.res, parasoft](#)
- ☐ **"context" may possibly be null** Why is this an issue? 4 minutes ago L29 [bd.except, parasoft](#)
- ☐ **"context" may possibly be null** Why is this an issue? 4 minutes ago L29 [bd.except, parasoft](#)

src/.../java/examples/flowanalysis/AlwaysCloseImages.java

- ☐ **Image resource not closed: imgReader** Why is this an issue? 4 minutes ago L16 [bd.res, parasoft](#)
- ☐ **Image resource not closed: imgReader** Why is this an issue? 4 minutes ago L16 [bd.res, parasoft](#)
- ☐ **"imgReader" may possibly be null** Why is this an issue? 4 minutes ago L31 [bd.except, parasoft](#)

View online: <https://docs.parasoft.com/x/z7K9Bw>

Viewing Test Execution Results

The screenshot displays the SonarQube web interface. At the top, the navigation bar includes 'sonarqube', 'Projects', 'Issues', 'Rules', 'Quality Profiles', 'Quality Gates', and 'Administration'. A search bar on the right contains the text 'Search for projects...'. Below the navigation bar, the project 'Demo Project' is selected, with a sub-menu showing 'master'. A status bar indicates 'Last analysis had 1 warning' on August 8, 2022, at 10:16 AM, with version 1.0.0.

The main content area is divided into three panels. The left panel, titled 'Issues', lists several security-related issues: 'Security context not disposed: GSSManager.getInstance().createCo...', 'context may possibly be null', and 'Image resource not closed: imgReader'. The middle panel, titled 'Code', shows the source code of the file '...amples/flowanalysis/AlwaysCloseGSS.java'. The code includes two methods: 'process' and 'processClose'. The right panel, titled 'Activity', displays a list of issues with details such as 'Security context not disposed: GSSManager.getInstance().createContext(tokens)', 'Why is this an issue?', '5 minutes ago', 'L13', and 'bd.res, parasoft'.

View online: <https://docs.parasoft.com/x/z7K9Bw>

Unit Test Execution and Coverage

Reporting test execution and test coverage should be done using the existing industry standard solutions supported by SonarQube. Other unit test and coverage frameworks are not currently supported.

1. Run unit tests to generate test execution and coverage reports.
2. Run SonarQube scanner to upload reports.

See <https://docs.sonarqube.org/latest/analysis/coverage/> for more details.