CWE Compliance

In this section:

Introduction

The Parasoft CWE Compliance artifact is a set of assets for your DTP infrastructure that enable you to track and visualize programming errors associated with CWE (Common Weakness Enumeration) guidelines. The artifact is shipped as part of the Security Compliance Pack. Contact your Parasoft representative for download and licensing information.

Supported Guidelines

The CWE Compliance artifact supports the following specific CWE implementations:

- 2019 CWE Top 25 Most Dangerous Software Errors
- CWE List Version 4.0 (Jtest and dotTEST only)
- CWE Top 25 + On the Cusp

Click on the following links to learn more about the supported CWE guidelines:

- http://cwe.mitre.org/top25/
- https://cwe.mitre.org/data/
- https://cwe.mitre.org/data/definitions/867.html

Prerequisites

The following Parasoft code analysis tools with appropriate Security Compliance licenses are supported:

- Jtest 2020.1 or later
- dotTEST 2020.1 or later
- C/C++test 2020.1 or later

Process Overview

1. Install the Security Compliance Pack into DTP Extension Designer.
2. Deploy the CWE Compliance artifact using Extension Designer. This action also deploys CWE Compliance assets to your DTP environment.
3. Connect your code analysis tool to your project in DTP. Configure the settings that enable DTP to correlate analysis results, i.e., build ID, source control settings, etc. See the documentation for your analysis tool for details.
4. Analyze the project with your code analysis tool using one of the CWE test configurations.
5. (Optional) Run the KPI workflow as part of your automated build process to generate metrics data associated with CWE compliance.
6. Use the DTP dashboard template, widgets, and reports to monitor compliance with security standards.

CWE Compliance Assets

The following artifacts are included in the package and added to your DTP environment when you install the Security Compliance Pack.

CWE Compliance.json

This is the core asset that extends DTP's data processing capabilities and produces CWE widgets and reports. DTP Workflows must be deployed using Extension Designer before they can be used (see Deploying the CWE Assets).

Test Configurations

You can configure your tool to run either the test configurations it ships with or the test configurations installed with the Security Compliance Pack. Refer to your tool's documentation for details. The following test configurations are included in the compliance pack:

- CWE 4.0 [Parasoft 2020.1].properties (Jtest and dotTEST only)
- CWE Top 25 2019 [Parasoft 2020.1].properties
- CWE Top 25 2019 + On the Cusp [Parasoft 2020.1].properties
- PCI DSS 3.2.properties [Parasoft 2020.1].properties (Jtest and dotTEST only)
- UL 2900 [Parasoft 2020.1].properties (Jtest and dotTEST only)

Dashboard Templates

Dashboard templates include preconfigured widgets to help you quickly view specific information about your projects. Refer to the Dashboards section to learn more about dashboards in DTP. See Adding the CWE Dashboards for details about viewing the widgets that appear in the dashboard templates.

The following template files are included in the CWE Compliance artifact:

- CWE 4.0 - .NET (CWE-4_0-dotNET.json)
- CWE 4.0 - Java (CWE-4_0-Java.json)
- CWE Top 25 2019 - .NET (CWE-Top-25-dotNET.json)
- CWE Top 25 2019 - Java (CWE-Top-25-Java.json)
- CWE Top 25 2019 - C/C++ (CWE-Top-25-Cpp.json)
- CWE Top 25 2019 + On the Cusp - .NET (CWE-Top-25-and-On-the-Cusp-dotNET.json)
- CWE Top 25 2019 + On the Cusp - Java (CWE-Top-25-and-On-the-Cusp-Java.json)
- CWE Top 25 2019 + On the Cusp - C/C++ (CWE-Top-25-and-On-the-Cusp-Cpp.json)

The Security Compliance pack ships with the following additional dashboard templates that include a combination of widgets configured to show CWE and OWASP compliance.

- UL 2900 - Java (ul2900-Java.json)
- UL 2900 - .NET (ul2900-dotNET.json)

Also see the OWASP Compliance documentation.

Models and Profiles

Profiles provide a range of functions in a DTP infrastructure, such as providing inputs for custom calculations executed by an extension and providing data for compliance reports. Profiles take their structure from models, which define fields, headers, or other components used in the profile. See Working with Model Profiles for information about understanding profiles in DTP Enterprise Pack.

The following profile files are included with the CWE artifact.

- CWE 4.0 - .NET profile (cwe-4_0-dotnet.json)
- CWE 4.0 - Java profile (cwe-4_0-java.json)
- CWE Security Impact - .NET profile (cwe-security-impact-dotnet.json)
- CWE Security Impact - Java profile (cwe-security-impact-java.json)
- CWE Security Impact - C++ (cwe-security-impact-cpp.json)
- CWE Top 25 - .NET profile (cwe-top25-2019-dotnet.json)
- CWE Top 25 - Java profile (cwe-top25-2019-java.json)
- CWE Top 25 - C++ (cwe-top25-2019-cpp.json)
- CWE Top 25+Cusp - .NET (cwe-top25-2019-on-the-cusp-dotnet)
- CWE Top 25+Cusp - Java (cwe-top25-2019-on-the-cusp-java)
- CWE Top 25+Cusp - C++ (cwe-top25-2019-on-the-cusp-cpp.json)
- CWE Compliance model (cwe-compliance.json)
- KPI model (KPI.json)

Compliance Categories

Individual code analysis rules belong to a category, such as Security, Exceptions, etc. The CWE Compliance artifact includes files that map code analysis rules to CWE-specific categories, i.e., weakness type or impact. You can configure widgets to report violations according to the categories defined in the following files to view them according to their CWE category:

- CWE 4.0 - Java (CWE-4_0-Java.xml)
- CWE Top 25 - Technical Impact - C++ (CWE-Top-25-Impact-Cpp.xml)
- CWE 4.0 - Software Development - Java (CWE-4_0-Software-Development-Java.xml)
- CWE Top 25 - Technical Impact - Java (CWE-Top-25-Impact-Java.xml)
- CWE 4.0 - Software Development - .NET (CWE-4_0-Software-Development-dotNET.xml)
- CWE 4.0 - Technical Impact - .NET (CWE-Top-25-Impact-dotNET.xml)
- CWE 4.0 - .NET (CWE-4_0-dotNET.xml)
- CWE Top 25+Cusp - C++ (CWE-Top-25-and-Cusp-Cpp.xml)
- CWE 4.0 - Technical Impact - Java (CWE-Impact-Java.xml)
- CWE Top 25+Cusp - Technical Impact - C++ (CWE-Top-25-and-Cusp-Impact-Cpp.xml)
- CWE Top 25 - Technical Impact - .NET (CWE-Impact-dotNET.xml)
- CWE Top 25+Cusp - Technical Impact - Java (CWE-Top-25-and-Cusp-Impact-Java.xml)
- CWE Top 25 - C++ (CWE-Top-25-2019-Cpp.xml)
- CWE Top 25+Cusp - Technical Impact - .NET (CWE-Top-25-and-Cusp-Impact-dotNET.xml)
- CWE Top 25 - Java (CWE-Top-25-2019-Java.xml)
- CWE Top 25+Cusp - Java (CWE-Top-25-and-Cusp-Java.xml)
- CWE Top 25+Cusp - Software Development - C++ (CWE-Top-25-2019-Software-Development-Cpp.xml)
- CWE Top 25 - Software Development - C++ (CWE-Top-25-and-Cusp-Software-Development-Cpp.xml)
- CWE Top 25 - Software Development - Java (CWE-Top-25-2019-Software-Development-Java.xml)

- CWE Top 25+Cusp - Software Development - Java (CWE-Top-25-and-Cusp-Software-Development-Java.xml)
- CWE Top 25 - Software Development - .NET (CWE-Top-25-2019-Software-Development-dotNET.xml)
- CWE Top 25+Cusp - Software Development - .NET (CWE-Top-25-and-Cusp-Software-Development-dotNET.xml)
- CWE Top 25 - .NET (CWE-Top-25-2019-dotNET.xml)
- CWE Top 25+Cusp - .NET (CWE-Top-25-and-Cusp-dotNET.xml)

See Custom Compliance Categories for additional information about rule categories in DTP.

Key Performance Indicator.json

This DTP Workflow performs additional calculations to provide metrics data specific to CWE. The KPI Workflow is optional and is not specific to the CWE Compliance artifact. To use this workflow, deploy it to your DTP environment and manually add instances of the standard Metrics - Summary widget to your dashboard to view the data. See Calculating Security Impact for details.

Cross-reference PDFs

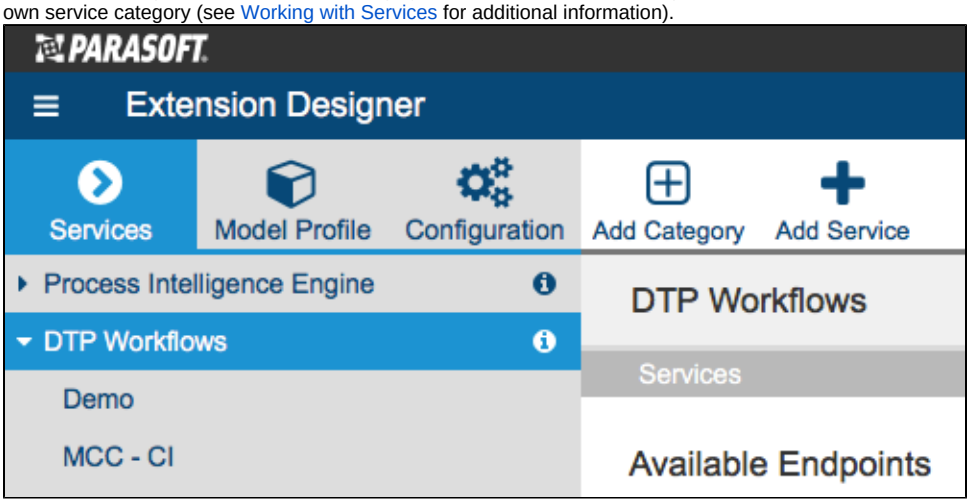
For your convenience, PDFs that show the association between Parasoft rules and CWE guidelines are located in the following directories:

- <PACK>/rules/jtest
- <PACK>/rules/dottest
- <PACK>/rules/cpptest

Deploying the CWE Assets

The CWE Compliance assets are installed when you install the Security Compliance Pack (see Installation). After installing the artifact, you must deploy the assets to your DTP environment.

1. Choose **Extension Designer** from the DTP settings (gear icon) menu.
2. Click the **Services** tab and expand the **DTP Workflows** services category. You can deploy assets under any service category you wish, but we recommend using the DTP Workflows category to match how Parasoft categorizes the assets. You can also click **Add Category** to create your own service category (see Working with Services for additional information).



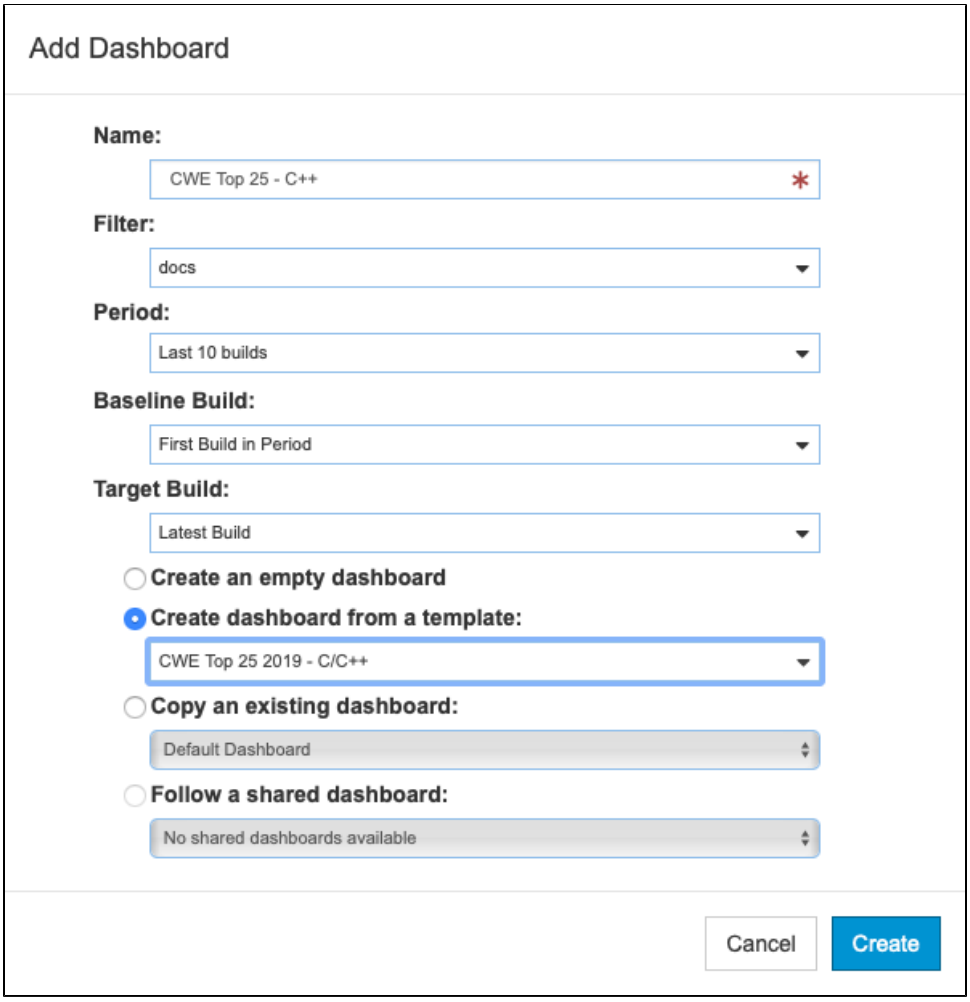
3. You can deploy the artifact to an existing service or add a new service. The number of artifacts deployed to a service affects the overall performance. See Extension Designer Best Practices for additional information. Choose an existing service and continue to step 5 or click **Add Service**.
4. Specify a name for the service and click **Confirm**.
5. The tabbed interface helps you keep artifacts organized within the service. Organizing your artifacts across one or more tabs does not affect the performance of the system. Click on a tab (or click the **+** button to add a new tab) and click the vertical ellipses menu.
6. Choose **Import> Library> Workflows> Security> CWE Compliance** and click anywhere in the open area to drop the artifact into the service.
7. Click **Deploy** and refresh your DTP browser window.

You can now add CWE widgets, use CWE compliance categories, and view CWE reports.

Adding the CWE Dashboards

The CWE Compliance dashboard templates will be available after installing the Security Compliance Pack. If you do not see the dashboard templates, restart DTP (see Stopping DTP Services and Starting DTP Services).

1. Click **Add Dashboard** in the DTP toolbar and specify a name when prompted.
2. (Optional) You can configure the default view for the dashboard by specifying the following information:
    - Choose the filter associated with your project from the Filter drop-down menu. A filter represents a set of run configurations that enabled custom views of the data stored in DTP. See Creating and Managing Filters for additional information.
    - Specify a range of time from the Period drop-down menu.
    - Specify a range of builds from the Baseline Build and Target Build drop-down menus.



3. Enable the **Create dashboard from a template** option and choose one of the CWE templates.
4. Click **Create** to finish adding the dashboard.

Repeat the process for any additional CWE dashboards you want to add to you DTP view.

If you have already executed your code analysis tool using the correlated CWE test configuration, widgets will render data as soon as the dashboard is added. You can immediately begin using these widgets and working with the data to help you track your compliance goals.

CWE Dashboards

See Dashboard Templates for a list of the dashboard templates shipped with the compliance artifact. The following widgets are included on one or more of the dashboards shipped with the Security Compliance pack:

CWE Compliance - Status

This widget shows the general compliance status of the project. It includes the build ID and the compliance category configuration used to display the results.

The widget can show the following states:

- Compliant - Code meets all required guidelines.
- Compliant with Deviations - Code meets all guidelines, but deviations have been applied. Deviations are violations that you have determined to be acceptable (see Deviation Report for additional information about deviations).
- Not Compliant - Code does not meet all required guidelines.
- Missing rule(s) in analysis - Parasoft code analysis rules documented in your profile were not included in the specified build.

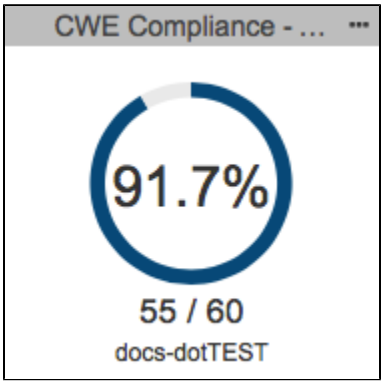
Click on the widget to open the CWE Compliance Report.

If you deploy the UL 2900 dashboard for Java or .NET, an OWASP Compliance - Status widget for OWASP Top 10 will also be included.

Click on the widget to open the OWASP Compliance Report (see OWASP Compliance for additional information).

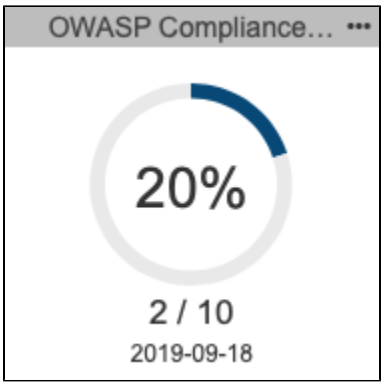
CWE Compliance - Percentage

This widget shows how much of the project is in compliance with the CWE guidelines.



Click on the widget to open the CWE Compliance Report.

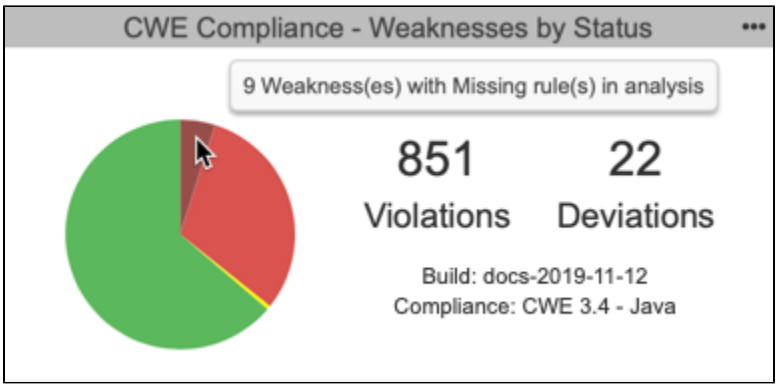
If you deploy the UL 2900 dashboard for Java or .NET, an OWASP Compliance - Percentage will also be included.



Click on the widget to open the OWASP Compliance Report (see OWASP Compliance for additional information).

CWE Compliance - Weakness by Status

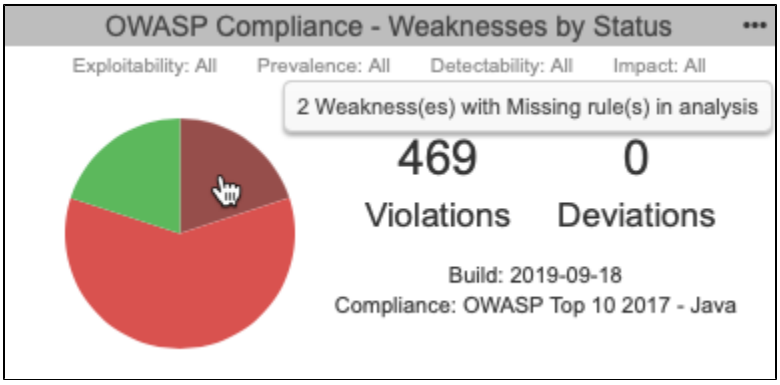
This widget shows the number of rules passed, violations, and deviations (suppressed code analysis violations). The green segment in the pie chart represents passing rules, while the red segment represents rules that have been violated. The widget also includes the build ID and the compliance category configuration used to display the results.



You can perform the following actions:

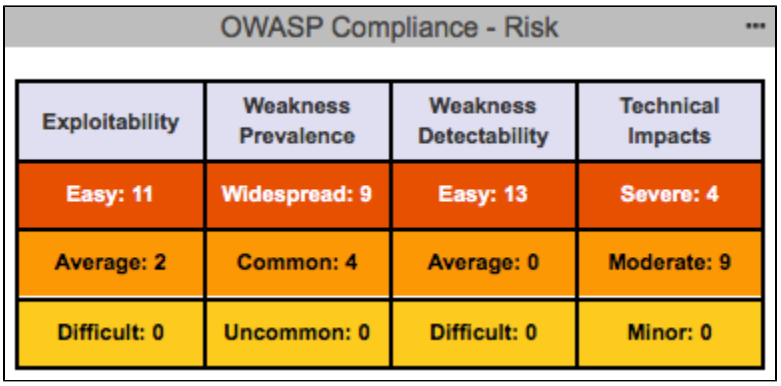
- Mouse over a segment of the pie chart to view details.
- Click on the passing segment of the pie chart to open the CWE Compliance Report filtered by passing guidelines.
- Click on the violations segment of the pie chart to open the CWE Compliance Report filtered by violations.
- Click on the Violations value to open an unfiltered instance of the CWE Compliance Report.
- Click on the Deviations value to open the Deviation Report.

The UL 2900 dashboard for Java or .NET includes an OWASP Compliance - Weakness by Status widget, as well.



OWASP Violations by Risk

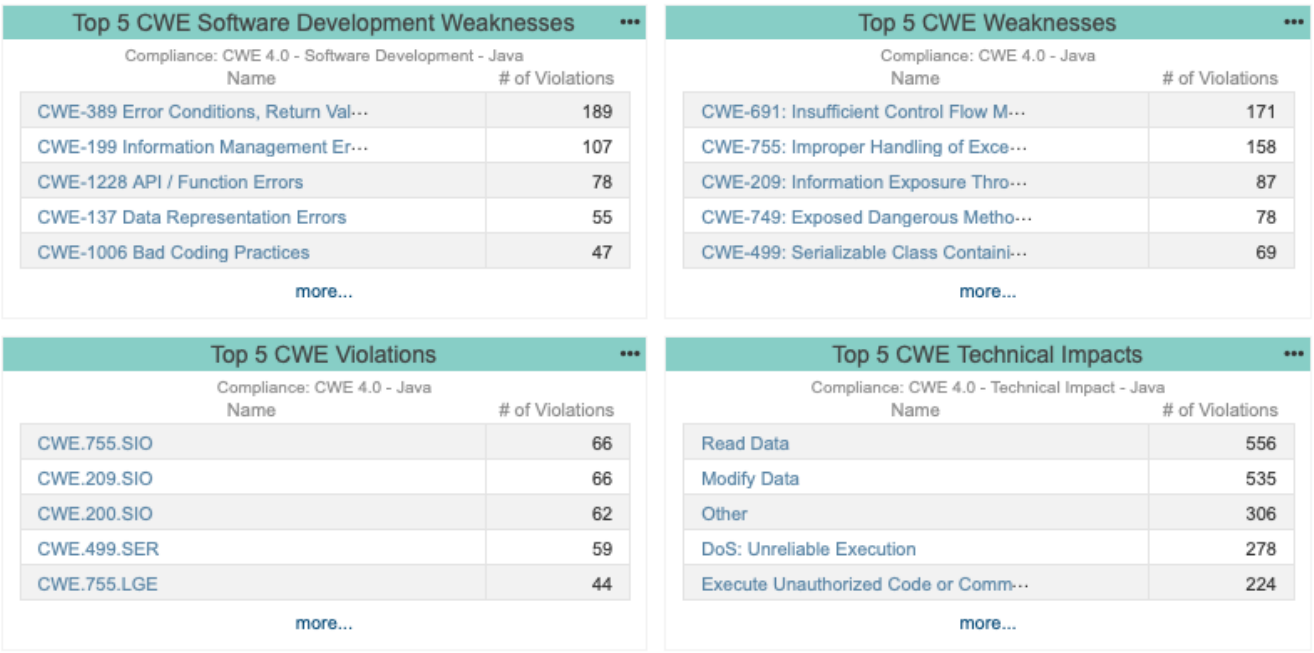
The UL 2900 dashboard for Java or .NET includes the OWASP Violation by Risk widget.



Refer to the OWASP Compliance documentation to learn more about this widget.

Violations by Category

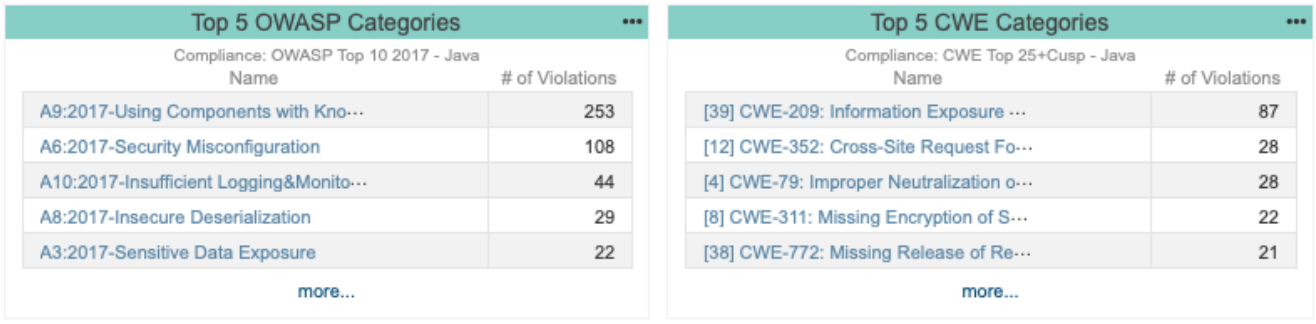
The dashboard includes several instances of the standard DTP Categories - Top 5 Table widget configured to show violations according to CWE guidelines.



Each instance of the widget is driven by the compliance category configuration (see Compliance Categories).

Click on a category link in the Name column to open the Violations by Rule report. Click on the **more...** link (if more than five categories contain violations) to view the Violations by Compliance Category report.

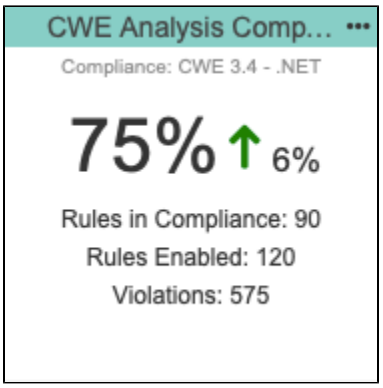
The UL 2900 dashboard for Java or .NET includes Categories - Top 5 Table widgets for CWE and OWASP.



The links in the UL 2900 dashboard widgets link to the same reports as the CWE dashboard widgets.

Rules in Compliance

The dashboard includes an instance of the standard DTP Rules in Compliance - Summary widget configured for CWE. This widget shows what percentage of the rules are in compliance, number of rules in compliance, rules enabled, and number of violations. Click on the widget to view the Violations by Compliance Category report.

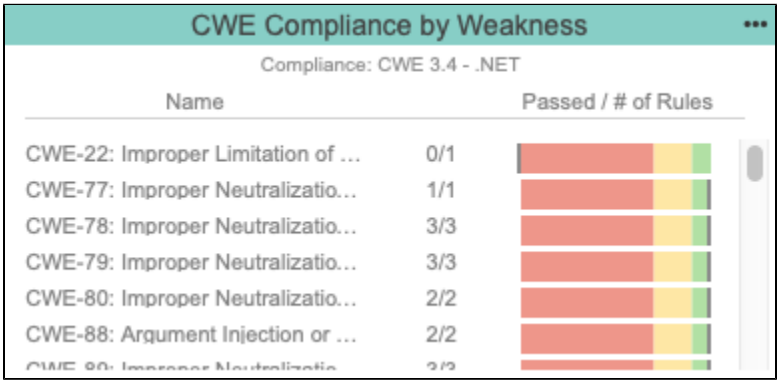


If you deploy the UL 2900 dashboard for Java or .NET, a Rules in Compliance widget configured for OWASP compliance will be included.

Click on the widget to view the OWASP Compliance Report (see OWASP Compliance for additional information).

Compliance by Category

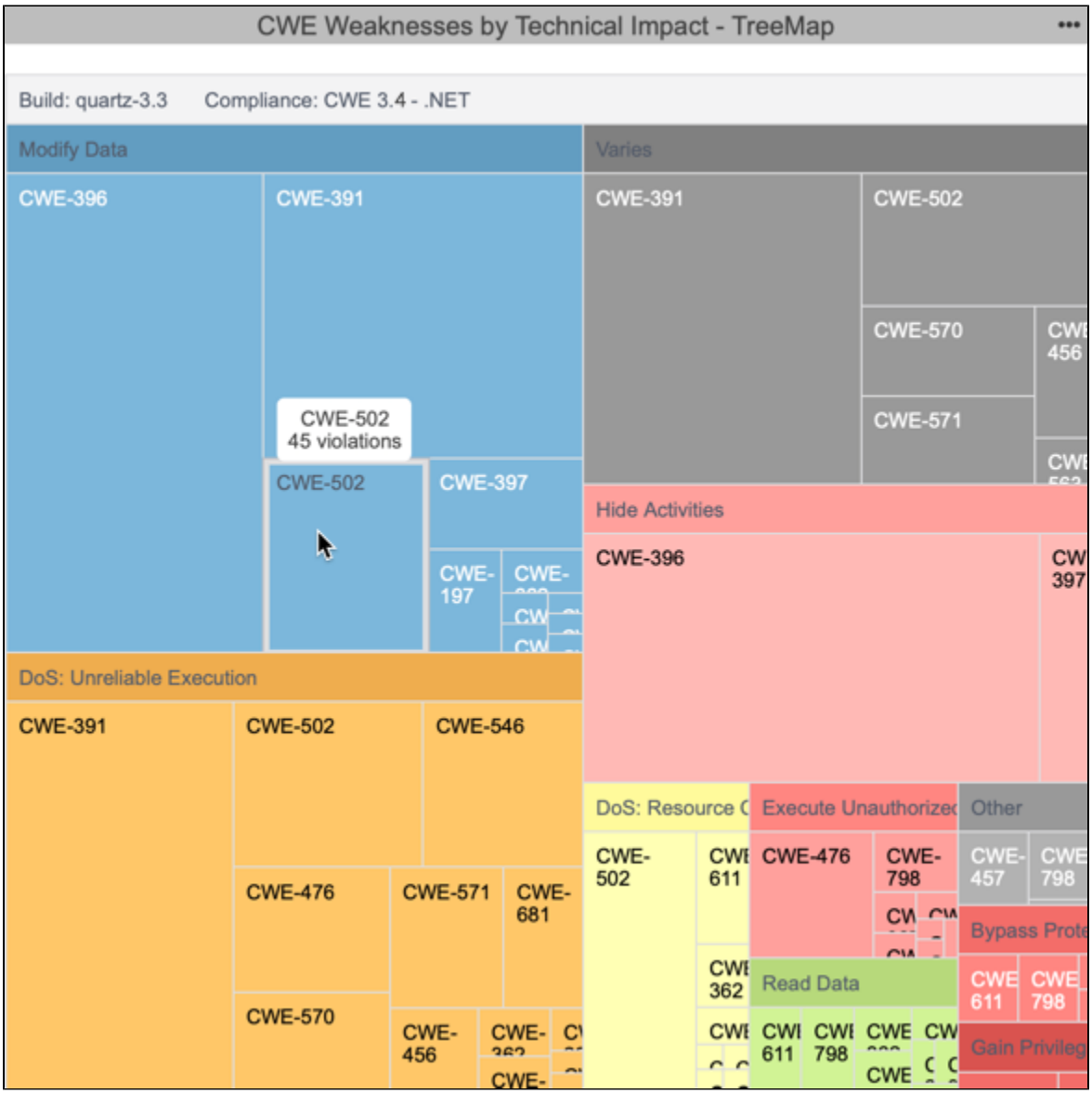
The dashboard includes an instance of the standard DTP Compliance By Category widget configured for CWE. This widget provides an overview of the compliance status for each category in the compliance configuration.



Click on the widget to open the Violations by Rule report.

CWE Weakness by Technical Impact - TreeMap

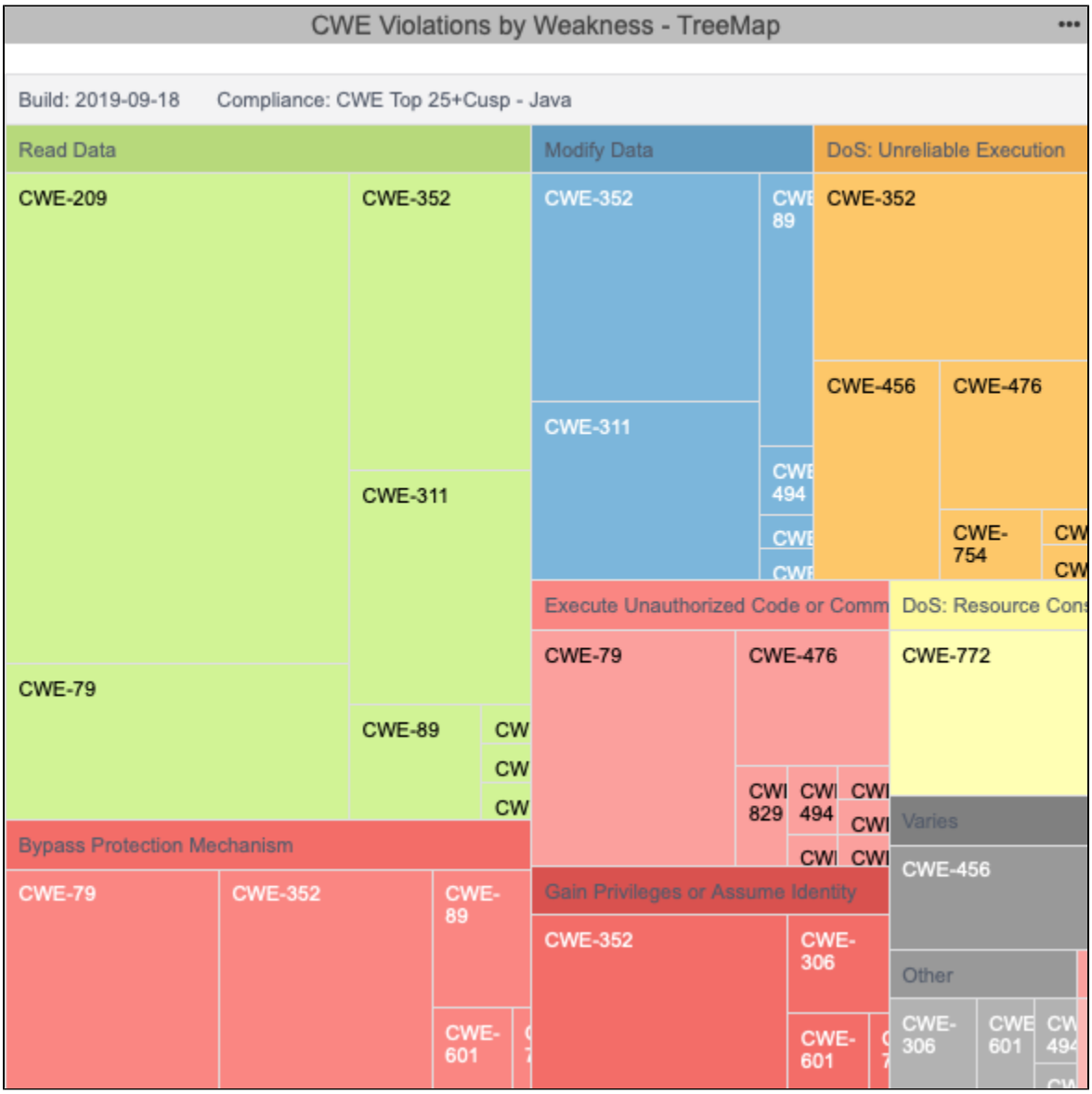
This widget shows how static analysis violations are concentrated according to their technical impact.

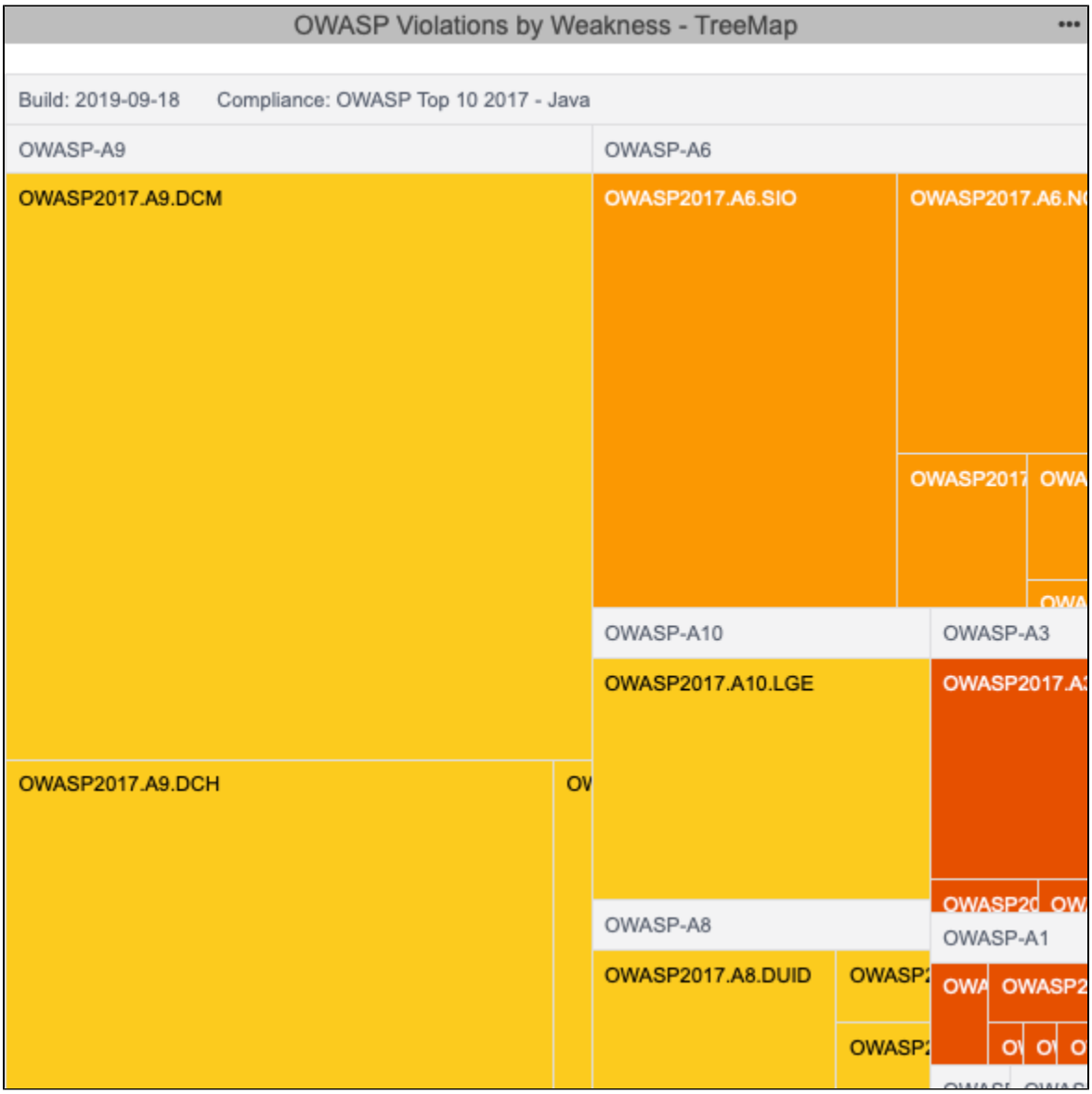


Mouse over a leaf in the widget to view details. Click on a leaf to open the Violations Explorer filtered by the compliance category.

Violations by Weakness - Treemap

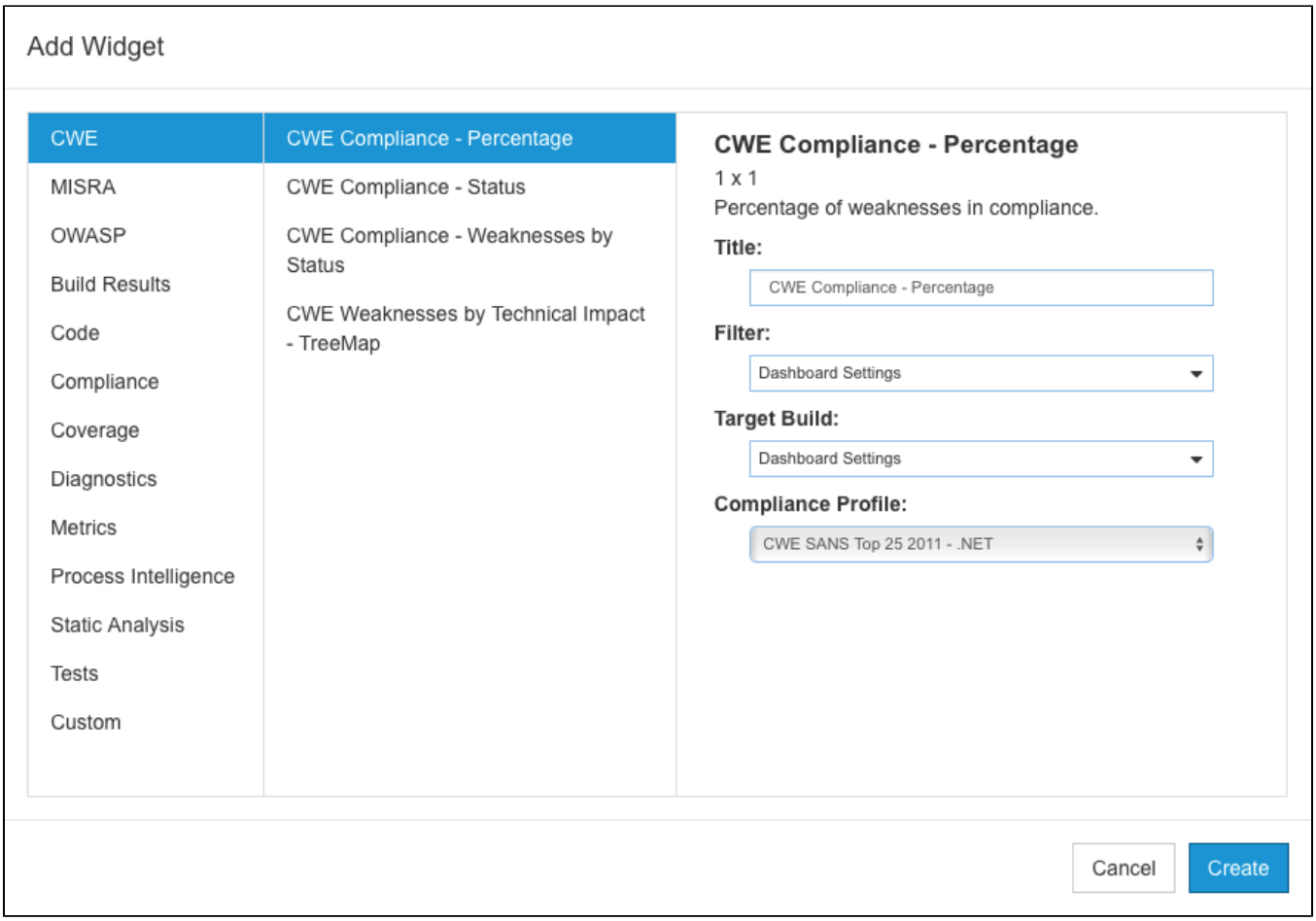
The UL 2900 dashboards for Java or .NET include tree map widgets for CWE and OWASP. Both widgets open the Violations Explorer.


CWE Violations by Weakness - TreeMap
Build: 2019-09-18
Compliance: CWE Top 25+Cusp - Java
Read Data
CWE-209
CWE-352
CWE-311
CWE-79
CWE-89
Modify Data
CWE-352
CWE-311
DoS: Unreliable Execution
CWE-352
CWE-456
CWE-476
CWE-754
Execute Unauthorized Code or Comm
CWE-79
CWE-476
DoS: Resource Cons
CWE-772
Varies
CWE-456
Other
Bypass Protection Mechanism
CWE-79
CWE-352
CWE-89
CWE-601
Gain Privileges or Assume Identity
CWE-352
CWE-306
CWE-601
CWE-306
CWE-601



Manually Adding the CWE Widgets

You can manually add the CWE widgets to an existing dashboard. See Adding Widgets for general instructions on how to add widgets to a dashboard. After deploying the artifact, widgets will appear in the CWE category.



CWE Widget Configuration Settings

Title	You can rename the widget in the Title field.
Filter	Choose a specific filter or Dashboard Settings from the drop-down menu. See Creating and Managing Filters for additional information. The filter should contain data that matches the type compliance profile you choose (Java, .NET, C++). For example, if the filter contains code analysis data on a .NET project then you should choose one of the .NET compliance profiles.
Target Build	Choose a specific build from the drop-down menu. The build selected for the entire dashboard is selected by default. See Using Build Administration for additional information about understanding builds. This setting is available for all widgets.
Compliance Profile	Choose a compliance profile from the drop-down menu to display the code analysis data against one of the supported CWE-specific sets of guidelines. You can choose one of the following profiles: • CWE 4.0 - .NET • CWE 4.0 - Java • CWE Top 25 - .NET • CWE Top 25 - Java • CWE Top 25 - C++ • CWE Top 25+Cusp - .NET • CWE Top 25+Cusp - Java • CWE Top 25+Cusp - C++ The type compliance profile (Java, .NET, C++) should match the data in the selected filter. For example, choose one of the .NET compliance profiles if the filter contains code analysis data on a .NET project.

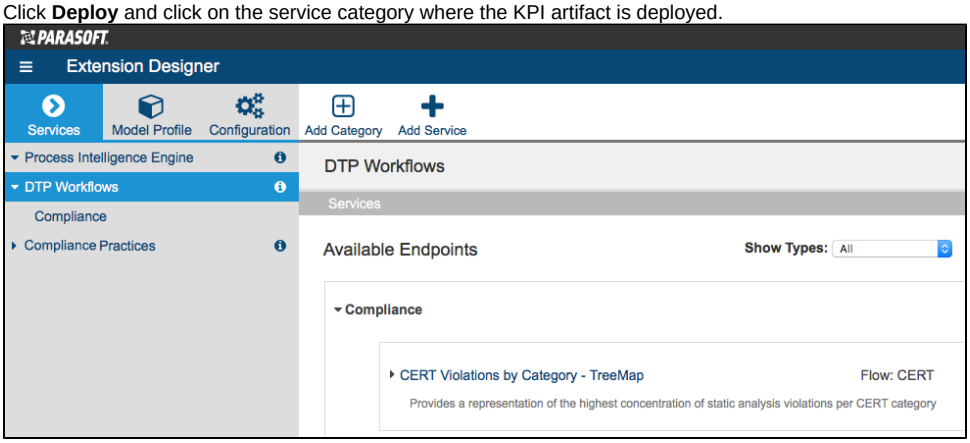
The CWE Compliance artifact includes profiles that you can use to calculate the security impact of detected weaknesses. Additional steps are required to leverage this functionality. See Calculating Security Impact for details.

Calculating Security Impact

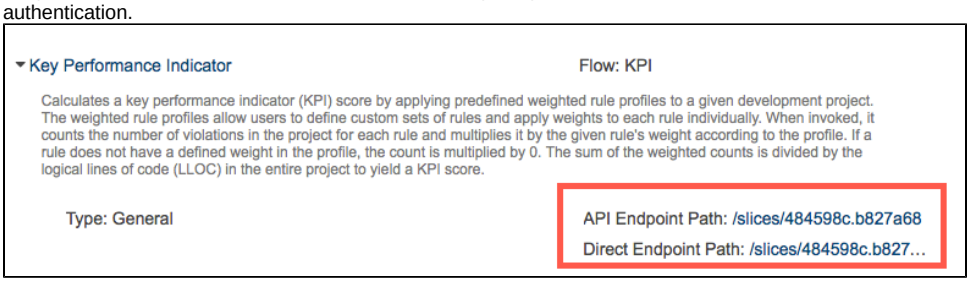
The Key Performance Indicator (KPI) DTP Workflow defines a KPI associated with static analysis rules so you can measure and quantify results. The build must have static analysis and metrics analysis data for the KPI extension to perform the calculation. The code analysis tool should have already been executed with the Metrics and a CWE test configuration test configurations under the same build ID. The metrics analysis must also include data for the Logical Lines of Code metric (metricId METRIC.NOLLOCIF). Refer to the tool documentation for details about setting the build ID and executing the Metrics test configuration.

This artifact needs to be deployed manually before you can use it.

1. Open **Extension Designer** and click on the **Services** tab.
2. Choose a service under a service category for running the KPI artifact. We recommend using a service in the DTP Workflows category to match how Parasoft categorizes the assets. You can deploy the artifact to an existing service or add a new service. The number of artifacts deployed to a service affects the overall performance.
3. The tabbed interface helps you keep artifacts organized within the service. Organizing your artifacts across one or more tabs does not affect the performance of the system. Click on a tab (or click the **+** button to add a new tab) and click the vertical ellipses menu.
4. Choose **Import> Library> Workflows> Security> Key Performance Indicator** and click anywhere in the open area to drop the artifact into the service.
5. Click **Deploy** and click on the service category where the KPI artifact is deployed.

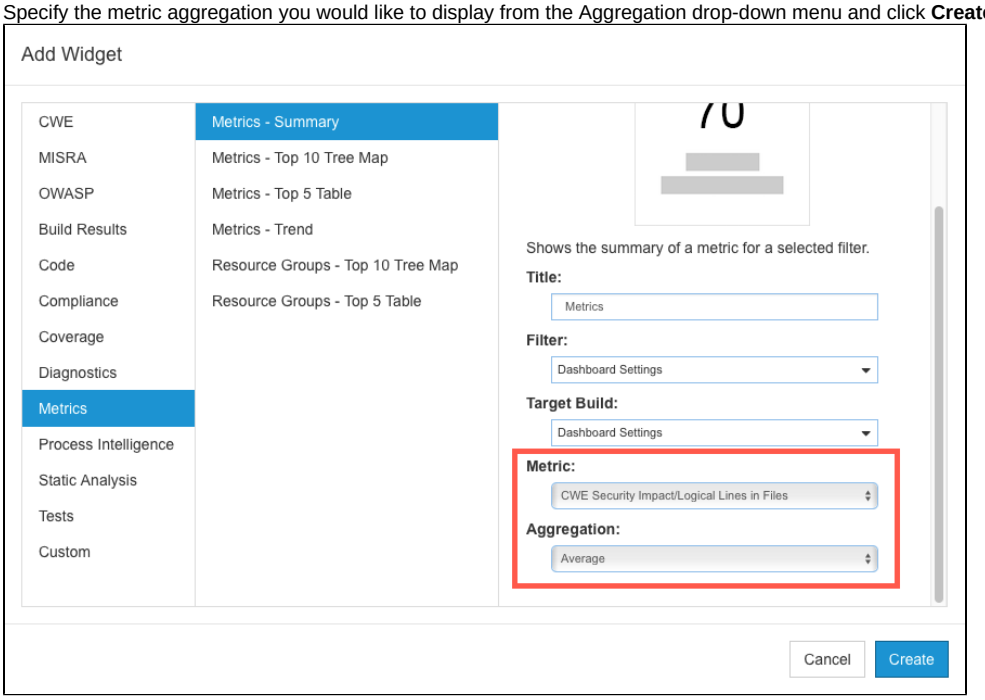


6. Expand the Key Performance Indicator section and copy the endpoint. Extension Designer presents two paths for the endpoint. The API Endpoint Path includes all API directories and can be used for exercising the endpoint in most cases. The Direct Endpoint Path is the direct path to the endpoint on the server and can be used if the API endpoint path is blocked or inaccessible, such as in some third-party integrations that require authentication.

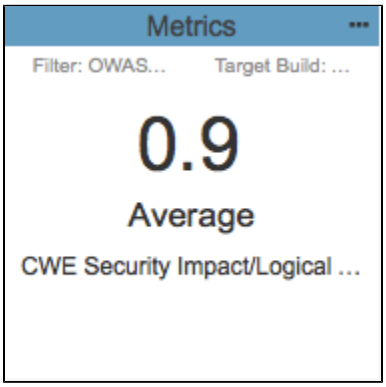


7. Send a REST request to the endpoint along with the required parameters. See Execution Details.
8. Open your DTP dashboard and click **Add Widget.**
9. Choose the **Metrics> Metrics - Summary** widget in the overlay.
10. Choose the **CWE Security Impact/Logical Lines in Files** from the Metric drop-down menu

11. Specify the metric aggregation you would like to display from the Aggregation drop-down menu and click **Create**.



The widget will appear on your dashboard.

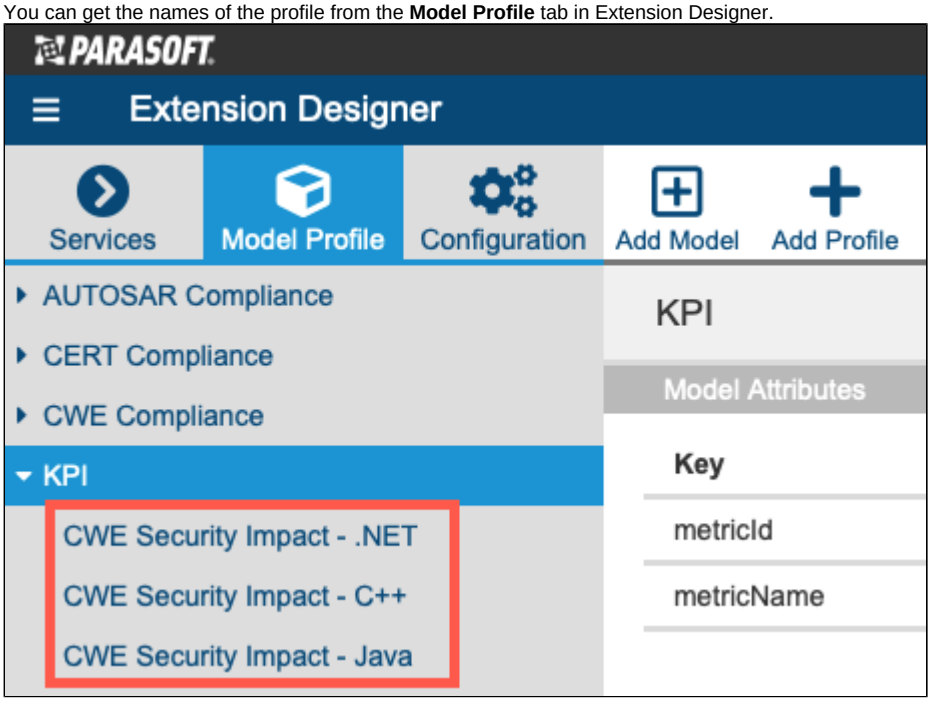


Clicking on the widget opens the Single Metric Overview Report.

Execution Details

You can execute the request in a browser, using a cURL command, or add it to a script. The following table describes the required parameters:

filterId	The filter ID for the project that the calculations will be performed on. You can quickly get the filter ID from URL of your dashboard.  You can also get the filter ID from the the Filters settings in DTP administration (see Creating and Managing Filters).
profile	The name of the profile that contains the rules and weights to for the calculation. Specify one of the following profiles to calculate security impact: • CWE Security Impact - .NET • CWE Security Impact - Java • CWE Security Impact - C++

	You can get the names of the profile from the **Model Profile** tab in Extension Designer. 
buildId	The build on which the calculation will be performed. If no build id is provided, this parameter defaults to the latest build.

Example API Call URL

```
http://framemaker.parasoft.com:8314/api/v1/services/5c0f0cae5d018e0630ae2789/slices/9acaecb1.7eb78?
filterId=9&profile=CWE%20Security%20Impact%20-%20.NET
```

Example Successful Response

```
{
        "success":{
                "title":"KPI",
                "message":"Calculation has started for filter 'CWE dotTEST' using profile 'CWE Security Impact
- .NET'. Check debug output for any errors during calculation."
        }
}
```

The default profile settings are based on the CWE standard, but you can remove or add rules, as well as change their default weights, in the profile. Creating custom profiles enables you to run different KPIs for different purposes and different profiles for different subsets of rules. You should preserve the default profiles and upload custom profiles to Extension Designer as necessary if you want to calculate custom KPIs. See Profiles for additional information.

Run KPI as part of your automated build process

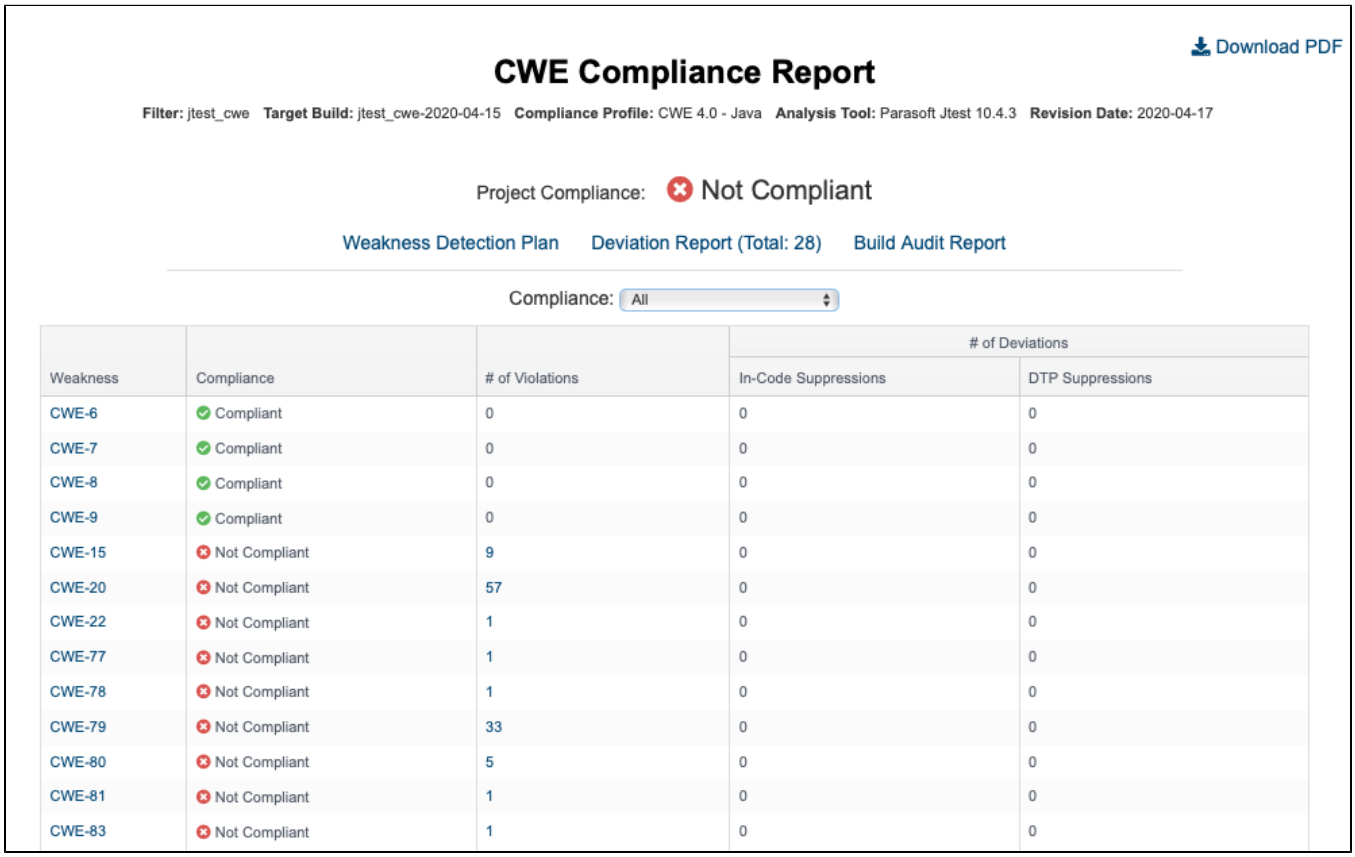
Depending on the volume of data being analyzed, KPI calculation may require multiple runs to acquire the core data and may take significant time, therefore triggering KPI calculation should be done as part of your build process or by manually using a trigger node in the KPI slice.

CWE Compliance Report

The CWE Compliance Report enables you to demonstrate compliance and monitor progress toward your compliance policy. The following CWE widgets link to the CWE Compliance Report:

- CWE Compliance - Status
- CWE Compliance - Percentage
- CWE Compliance - Weakness by Status

The report includes data for the build ID and filter configured in the widget you clicked to access the report. The compliance status of the project is also determined by the compliance profile configuration specified in the widget you clicked to access the report (see CWE Widget Configuration Settings).



Download PDF

CWE Compliance Report

Filter: jtest_cwe **Target Build:** jtest_cwe-2020-04-15 **Compliance Profile:** CWE 4.0 - Java **Analysis Tool:** Parasoft Jtest 10.4.3 **Revision Date:** 2020-04-17

Project Compliance: Not Compliant

Weakness Detection Plan | Deviation Report (Total: 28) | Build Audit Report

Compliance: All

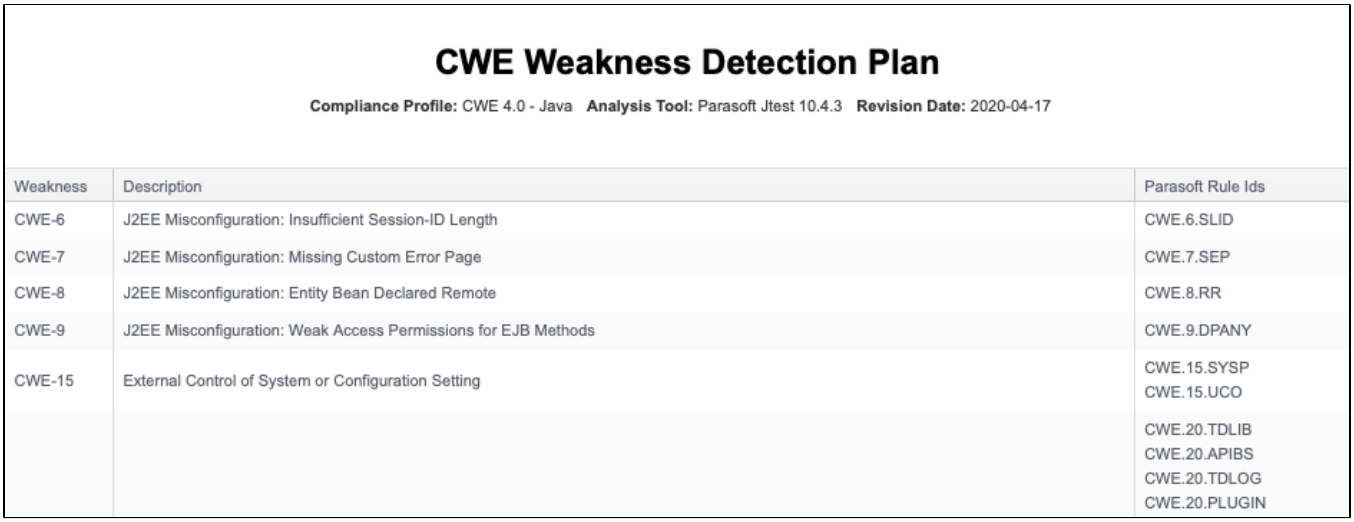
Weakness	Compliance	# of Violations	# of Deviations	
			In-Code Suppressions	DTP Suppressions
CWE-6	Compliant	0	0	0
CWE-7	Compliant	0	0	0
CWE-8	Compliant	0	0	0
CWE-9	Compliant	0	0	0
CWE-15	Not Compliant	9	0	0
CWE-20	Not Compliant	57	0	0
CWE-22	Not Compliant	1	0	0
CWE-77	Not Compliant	1	0	0
CWE-78	Not Compliant	1	0	0
CWE-79	Not Compliant	33	0	0
CWE-80	Not Compliant	5	0	0
CWE-81	Not Compliant	1	0	0
CWE-83	Not Compliant	1	0	0

You can perform the following actions:

- Click on one of the following links to open a sub-report:
  - Weakness Detection Plan
  - Deviation Report
  - Build Audit Report
- Choose a state from the Compliance drop-down menu to filter weaknesses by their current state.
- Click on a column header to sort the report.
- Click on a link in the Weakness column to go directly to the weakness in the Weakness Detection Plan report.
- Click on a value in the # of Violations column to view the violations in the Violations Explorer.
- Click on a value in one of the # of Deviations columns to view the suppressed violation in the Violations Explorer.
- Click **Download PDF** to export a printer-friendly PDF version of the report data. If you added a custom graphic to DTP as described in Adding a Custom Graphic to the Navigation Bar, the PDF will also be branded with the graphic.

Weakness Detection Plan

The Weakness Detection Plan shows how Parasoft code analysis rules map to weaknesses. This report is populated with data from the selected compliance profile (see Models and Profiles).



CWE Weakness Detection Plan

Compliance Profile: CWE 4.0 - Java **Analysis Tool:** Parasoft Jtest 10.4.3 **Revision Date:** 2020-04-17

Weakness	Description	Parasoft Rule Ids
CWE-6	J2EE Misconfiguration: Insufficient Session-ID Length	CWE.6.SLID
CWE-7	J2EE Misconfiguration: Missing Custom Error Page	CWE.7.SEP
CWE-8	J2EE Misconfiguration: Entity Bean Declared Remote	CWE.8.RR
CWE-9	J2EE Misconfiguration: Weak Access Permissions for EJB Methods	CWE.9.DPANY
CWE-15	External Control of System or Configuration Setting	CWE.15.SYSP CWE.15.UCO
		CWE.20.TDLIB CWE.20.APIBS CWE.20.TDLOG CWE.20.PLUGIN

Deviation Report

The Deviation Report shows information about which violations have been suppressed in the project. See Suppressing Violations for information about suppressions in DTP. Refer to the documentation for your analysis tool to learn about in-code suppressions.

CWE Deviation Report

Filter: jtest_cwe **Target Build:** cwe-3.4 **Compliance Profile:** CWE 3.4 - Java **Analysis Tool:** Parasoft Jtest 10.4.3 **Revision Date:** 2020-03-20

Only Deviations: ☐ Hide Modification History: ☐

CWE-6 J2EE Misconfiguration: Insufficient Session-ID Length ✔ - No Deviations

CWE-7 J2EE Misconfiguration: Missing Custom Error Page ✔ - No Deviations

CWE-8 J2EE Misconfiguration: Entity Bean Declared Remote ✔ - No Deviations

CWE-9 J2EE Misconfiguration: Weak Access Permissions for EJB Methods ✔ - No Deviations

CWE-15 External Control of System or Configuration Setting ✔ - No Deviations

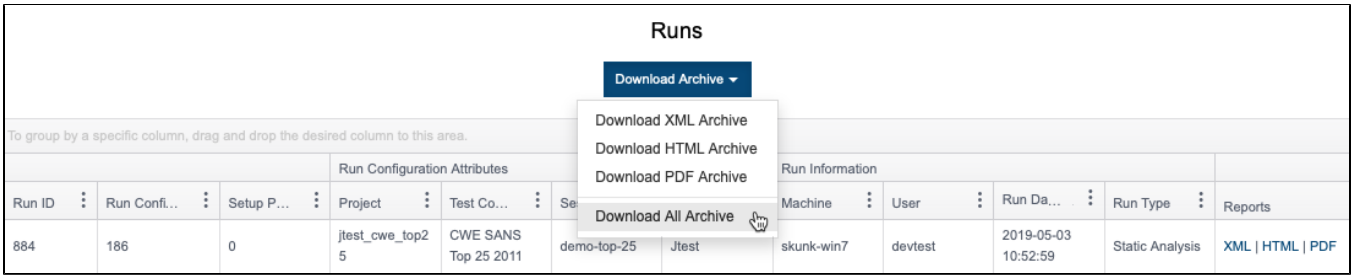
CWE-20 Improper Input Validation ! - 4 Deviations

Violation ID: d65c60cd-faa9-3990-8483-7afaaa602cfd
File: com.parasoft:demo/src/main/java/examples/flowanalysis/SystemInjection.java
Line: 42
Rule ID: CWE.20.APIBS
Deviation Type: In-Code Suppression
Action: None
Risk/Impact: Undefined
Suppression Reason: Library injection is expected here.
Suppression Author: Developer007

By default, the report shows all guidelines, but you can enable the **Only Deviations** option to filter out guidelines that have no suppressions associated with them. You can also enable the **Hide Modification History** option to exclude the modification history for deviations.

Build Audit Report

The Build Audit Report is native functionality in DTP. It shows an overview of code analysis violations, as well as test results and coverage information, associated with the build. This report also allows you to download an archive of the data, which is an artifact you can use to demonstrate compliance with CWE during a regulatory audit.



In order to download an archive, the build has to be locked. See Build Audit Report for additional details.

Profiles

Profiles provide additional inputs that enable custom calculations. The Security Compliance Pack includes a set of profiles that enable the data to viewed in the context of CWE standards. See Models and Profiles for list of the profiles used for CWE compliance. You can create custom profiles if you want to customize how DTP reports CWE data.

CWE Compliance Profiles

The default profiles show the correlation between CWE guidelines and Parasoft code analysis rules and are suitable for most normal use cases.

> **Do not modify the CWE profiles**
>
> We strongly advise you to avoid changing the default CWE profiles because doing so will affect any reports you may need to generate for auditing purposes.

If necessary, you can make a copy of the default profile and adjust the correlation between Parasoft code analysis rules and CWE guidelines to achieve your software quality and compliance goals.

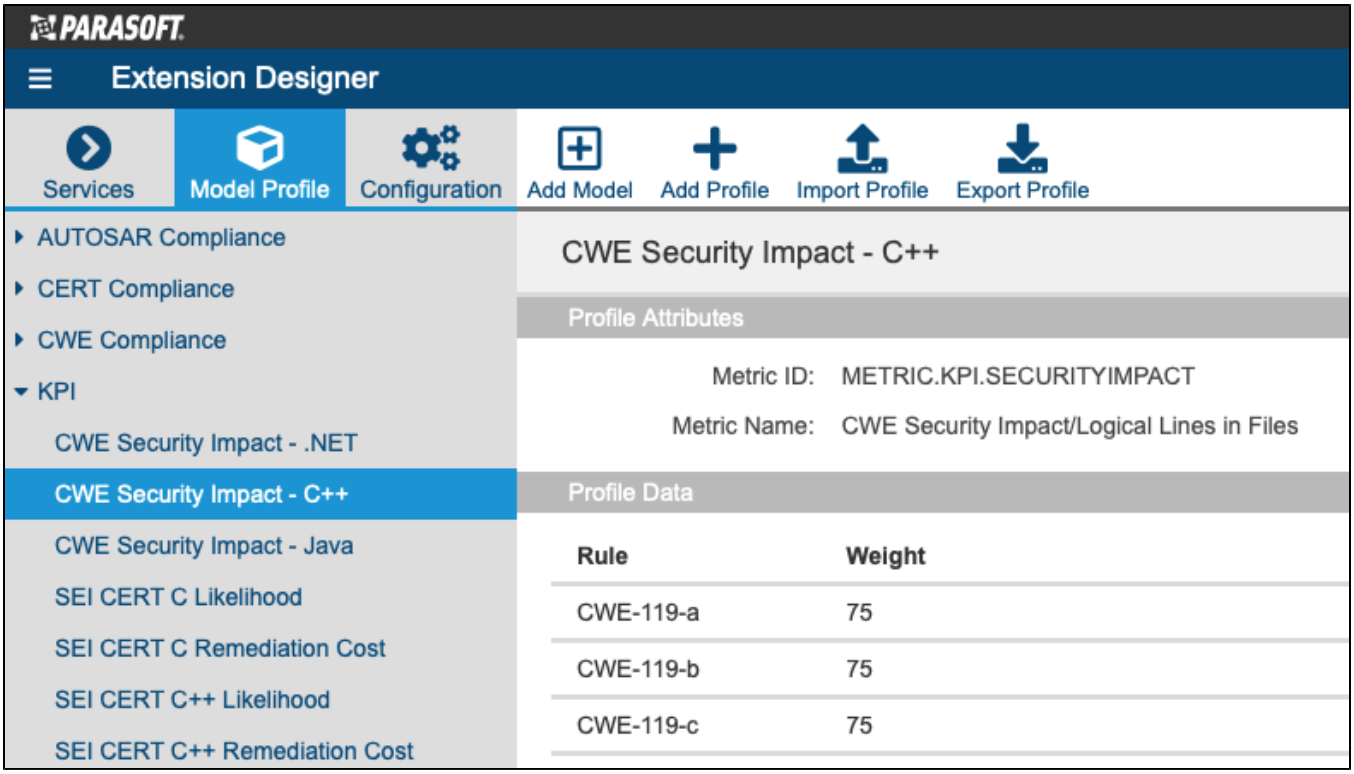
1. Open Extension Designer and click the **Model Profile** tab.
2. Expand the CWE Compliance model and choose one of the profiles.
3. Click **Export Profile** to download a copy.
4. Click **Add Profile** and enter a name.
5. Click **Confirm** to create an empty profile.
6. Rename the copy of the default profile you exported and click **Import Profile**.
7. Browse for the copy and confirm to upload.
8. Click **Edit** and make your adjustments.
9. Click **Save.**

CWE KPI Profiles

The KPI artifact shipped with the Security Compliance Pack includes the following profiles:

- CWE Security Impact - .NET
- CWE. Security Impact - C++
- CWE Security Impact - Java

The profiles assign weights to the metrics analysis rules in order to calculate a KPI value for the build.



The default profile is suitable for most normal usage, but you can adjust the weights for each metrics rule if necessary.

1. Open Extension Designer and click the **Model Profile** tab.
2. Expand the KPI model and choose a profile.
3. Click **Export Profile** to download a copy.
4. Click **Add Profile** and enter a name.
5. Click **Confirm** to create an empty profile.
6. Rename the copy of the default profile you exported and click **Import Profile**.
7. Browse for the copy and confirm to upload.
8. Click **Edit** and make your adjustments.
9. Click **Save.**