OpenText™ Application Security Analysis Extension for Visual Studio Code

User Guide

Version : 25.4.0

PDF Generated on : December 19, 2025

© Copyright 2025 Open Text

Table of Contents

1. User Guide

Software Version: 25.4.0

Document Release Date: December 2025

Software Release Date: December 2025

1.1. Change Log

The following table lists changes made to this document. Revisions to this document are published between software releases only if the changes made affect product functionality.

Software Release / Document Version	Changes
25.4.0	Added • Renamed the extension to OpenText™ Application Security Analysis Extension for Visual Studio Code. • Incorporate product name changes. () • Improved user interface and analysis workflow of OpenText™ Application Security Analysis Extension for Visual Studio Code.
23.1.0	Updated: • This document has been updated to include the new Fortify Remediation Extension for Visual Studio Code that you can use to audit and remediate your code by reviewing analysis results on Application Security from VS Code (see Remediating your Code) • The Fortify Extension for Visual Studio was updated to include a link to the Fortify Remediation Extension for Visual Studio Code (see Opening Application Security Application Versions)
22.1.0	Updated: • Minor edits

1.2. Getting Started

This document describes how to install and use:

- The OpenText™ Application Security Analysis Extension for Visual Studio Code to analyze your project with OpenText SAST to uncover any security issues.

  You can analyze your project with a locally installed OpenText SAST, upload the project to OpenText™ Core Application Security, or upload the project to ScanCentral SAST. When you analyze your project with OpenText SAST, you have the option to upload the analysis results to Application Security. You can analyze your project remotely using ScanCentral SAST and then upload the analysis results to Application Security.

- The Fortify Remediation Extension for Visual Studio Code to review analysis results from Application Security so you can resolve security-related issues in VS Code.

This section contains the following topics:

1.2.1. Product name changes

OpenText is in the process of changing the following product names:

Previous name	New name
Fortify Static Code Analyzer	OpenText™ Static Application Security Testing (OpenText SAST)
Fortify Software Security Center	OpenText™ Application Security
Fortify ScanCentral SAST	OpenText™ ScanCentral SAST
Fortify on Demand	OpenText™ Core Application Security
Debricked	OpenText™ Core Software Composition Analysis (OpenText Core SCA)
Fortify Applications and Tools	OpenText™ Application Security Tools

The product names have changed on product splash pages, mastheads, login pages, and other places where the product is identified. The name changes are intended to clarify product functionality and to better align the Fortify Software products with OpenText. In some cases, such as on the documentation title page, the old name might temporarily be included in parenthesis. You can expect to see more changes in future product releases.

1.2.2. Software Requirements

This topic describes the OpenText Application Security Software that the Application Security extensions for VS Code work with and the requirements for each task.

OpenText™ Application Security Analysis Extension for Visual Studio Code Requirements

To analyze your code, make sure the following requirements are met depending on the type of analysis you are using.

Software	Version	Requirements
OpenText™ Core Application Security	N/A	To upload your project to OpenText™ Core Application Security for assessment, make sure that you have the following: • ScanCentral SAST standalone client installed and included in the PATH environment variable • OpenText™ Core Application Security credentials.

Software	Version	Requirements
OpenText SAST	24.2.0 or later	To scan your project locally with OpenText SAST, you must either: • Make sure that the PATH environment variable includes the sourceanalyzer executable • Have the full path to the OpenText SAST installation directory Make sure that your system meets the system requirements for the OpenText SAST version you are using as described in the *OpenText™ Application Security System Requirements* document in Fortify Static Code Analyzer and Tools Documentation.

Software	Version	Requirements
ScanCentral SAST	24.2.0 or later	To scan your project remotely with ScanCentral SAST, make sure that you have one of the following: • A Application Security URL. • An authentication token of type `ToolsConnectToken`. For languages that are supported for analysis and system requirements for the ScanCentral SAST version you are using, see the *OpenText™ Application Security System Requirements* document in Fortify Static Code Analyzer and Tools Documentation.

Software	Version	Requirements
Application Security	24.2.0 or later	To upload analysis results to Application Security after an analysis with ScanCentral SAST, make sure you have the following: • An application version that exists in Application Security • An authentication token of type `ToolsConnectToken`.

Fortify Remediation Extension for Visual Studio Code Requirements

To open analysis results on Application Security and audit and remediate your code, you must have the following:

- An Application Security server URL.
- The Application Security version must correspond with the Fortify Remediation Extension for Visual Studio Code version. The version number format is *<major>.<minor>.<patch>* (for example, 23.1.0). The *<major>* and *<minor>* portions of the Application Security and the Fortify Remediation Extension for Visual Studio Code version numbers must match. For example, versions 23.1.0 and 23.1.1 correspond.
- A user account on the Application Security server that has permission to access application versions.

  To log into Application Security, you can use a user name and password or an authentication token of type `ToolsConnectToken`.
- To audit issues in the analysis results, your user account must have audit permissions.

- To add comments to issues or assign custom tags that require comments, your user account must have the permission to comment on issues.

1.2.3. Installing the Fortify Extensions for Visual Studio Code

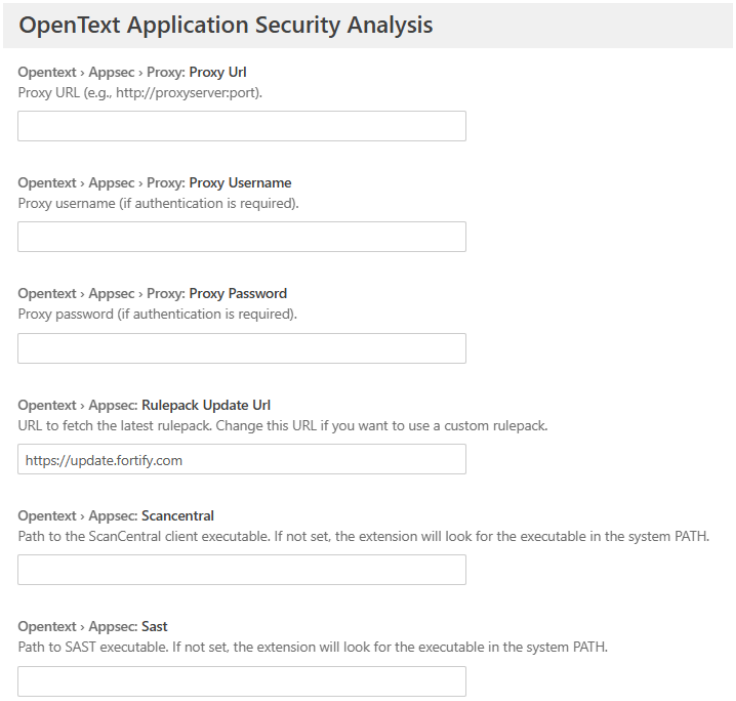
You can install the extensions on a computer running Windows, Linux, or macOS. Install either extension from the Visual Studio Marketplace. See the Visual Studio Code documentation for instructions about how to install an extension.

You can install the extension that best fits your needs or install both extensions.

1.2.4. Configuring the Application Security Analysis Extension for Visual Studio Code

To configure the OpenText™ Application Security Analysis Extension for Visual Studio Code extension settings:

1. Open VS Code **Settings** and search for `OpenText Application Security Analysis`.



2. Configure the following properties for OpenText Application Security Analysis:

Property	Description
Proxy: Proxy Url	Enter the proxy server URL for Application Security.  **Example** http://my.domain.com:8080/ssc
Proxy: Proxy Username	If HTTPS authentication is required, type a user name.
Proxy: Proxy Password	If HTTPS authentication is required, type a password.
Appsec: Rulepack Update Url	To acquire the latest Application Security Rulepacks, type a Rulepack update server URL. You can type the Application Security URL or the Fortify update server. Default value: https://update.fortify.com

Property	Description
Appsec: ScanCentral	Specifies the file path to ScanCentral SAST client executable. Go to the ScanCentral SAST installation directory and do one of the following: ◦ If you are using a standalone client installed with OpenText™ Application Security Tools, navigate to `<tools_install_dir>/bin/` and select `scancentral.bat` (on Windows) or `scancentral` (on non-Windows). ◦ If the standalone client is installed in a different location, navigate to the installation directory and select `scancentral.bat` (on Windows) or `scancentral` (on non-Windows). If you do not configure this property, the extension searches the ScanCentral SAST client executable path in the system PATH environment variable.
Appsec: Sast	Specifies the path to the locally installed OpenText SAST executable (`sourceanalyzer.exe`). If you do not configure this property, the extension searches the path to the `sourceanalyzer.exe` in the system PATH environment variabl

1.2.5. Related Documents

This topic describes documents that provide information about OpenText™ Application Security software products.

All Products

The following documents provide general information for all products. Unless otherwise noted, these documents are available on the Product Documentation website for each product.

Document / File Name	Description
About OpenText Application Security Software Documentation appsec-docs-n-*<version>*.pdf	This paper provides information about how to access OpenText Application Security Software product documentation. **Note** This document is included only with the product download.
OpenText™ Application Security System Requirements appsec-sr-*<version>*.pdf	This document provides the details about the environments and products supported for this version of OpenText™ Application Security Software.
OpenText™ Application Security *Software Release Notes* appsec-rn-*<version>*.pdf	This document provides an overview of the changes made to OpenText™ Application Security Software for this release and important information not included elsewhere in the product documentation.
What's New in OpenText™ Application Security *Software <version>* appsec-wn-*<version>*.pdf	This document describes the new features in OpenText™ Application Security Software products.

OpenText™ ScanCentral SAST

The following document provides information about OpenText™ ScanCentral SAST. Unless otherwise noted, this document is available on the Product Documentation website at https://www.microfocus.com/documentation/fortify-software-security-center.

Document / File Name	Description
OpenText™ ScanCentral SAST Installation, Configuration, and Usage Guide sc-sast-ugd-*<version>*.pdf	This document provides information about how to install, configure, and use ScanCentral SAST to streamline the static code analysis process. It is written for anyone who intends to install, configure, or use ScanCentral SAST to offload the resource-intensive translation and scanning phases of their OpenText SAST process.

OpenText™ Application Security

The following document provides information about OpenText™ Application Security. Unless otherwise noted, this document is available on the Product Documentation website at https://www.microfocus.com/documentation/fortify-software-security-center.

Document / File Name	Description
OpenText™ Application Security User Guide ssc-ugd-*<version>*.pdf	This document provides OpenText™ Application Security users with detailed information about how to deploy and use Application Security. It provides all of the information you need to acquire, install, configure, and use Application Security. It is intended for use by system and instance administrators, database administrators (DBAs), enterprise security leads, development team managers, and developers. Application Security provides security team leads with a high-level overview of the history and current status of a project.

OpenText SAST

The following documents provide information about OpenText SAST. Unless otherwise noted, these documents are available on the Product Documentation website at https://www.microfocus.com/documentation/fortify-static-code-analyzer-and-tools.

Document / File Name	Description
OpenText™ Static Application Security Testing User Guide sast-ugd-*<version>*.pdf	This document describes how to install and use OpenText SAST to scan code on many of the major programming platforms. It is intended for people responsible for security audits and secure coding.
OpenText™ Static Application Security Testing Custom Rules Guide sast-cr-ugd-*<version>*.zip	This document provides the information that you need to create custom rules for OpenText SAST. This guide includes examples that apply rule-writing concepts to real-world security issues. **Note** This document is included only with the product download.
Fortify License and Infrastructure Manager Installation and Usage Guide lim-ugd-*<version>*.pdf	This document describes how to install, configure, and use the Fortify License and Infrastructure Manager (LIM), which is available for installation on a local Windows server and as a container image on the Docker platform.

OpenText™ Application Security Tools

The following documents provide information about OpenText SAST applications and tools. Unless otherwise noted, these documents are available on the Product

Documentation website at https://www.microfocus.com/documentation/fortify-static-code-analyzer-and-tools.

Document / File Name	Description
OpenText™ Application Security Tools Guide sast-tgd-<*version*>.pdf	This document describes how to install OpenText™ Application Security Tools. It provides an overview of the applications and command-line tools that enable you to scan your code with OpenText SAST, review analysis results, work with analysis results files, and more.
OpenText™ Fortify Audit Workbench User Guide awb-ugd-<*version*>.pdf	This document describes how to use Fortify Audit Workbench to scan software projects and audit analysis results. This guide also includes how to integrate with bug trackers, produce reports, and perform collaborative auditing.
OpenText™ Fortify Plugin for Eclipse User Guide ep-udg-<*version*>.pdf	This document provides information about how to install and use the Fortify Complete Plugin for Eclipse.
OpenText™ Fortify Analysis Plugin for IntelliJ IDEA and Android Studio User Guide iap-udg-<*version*>.pdf	This document describes how to install and use Fortify Analysis Plugin for IntelliJ IDEA and Android Studio.
OpenText™ Fortify Extension for Visual Studio User Guide vse-ugd-<*version*>.pdf	This document provides information about how to install and use the Fortify extension for Visual Studio to analyze, audit, and remediate your code to resolve security-related issues in solutions and projects.

1.3. Analyzing your Code

The OpenText™ Application Security Extensions for Visual Studio Code provides three ways to analyze your source code to detect security vulnerabilities.

- Perform a static assessment, static+ assessment, or open-source scan for an application release on OpenText™ Core Application Security.
- Run a locally installed version of OpenText SAST on the currently opened project. To view the analysis results, upload the Fortify Project Results (FPR) file to an Application Security server.
- Run a remote analysis using ScanCentral SAST.

You can also perform a quick scan on the currently opened project and upload the analysis results to OpenText™ Core Application Security or Application Security

The following sections describe any prerequisites for each analysis method and the instructions for how to use it.

This section contains the following topics:

1.3.1. Uploading Code to OpenText Core Application Security for Assessment

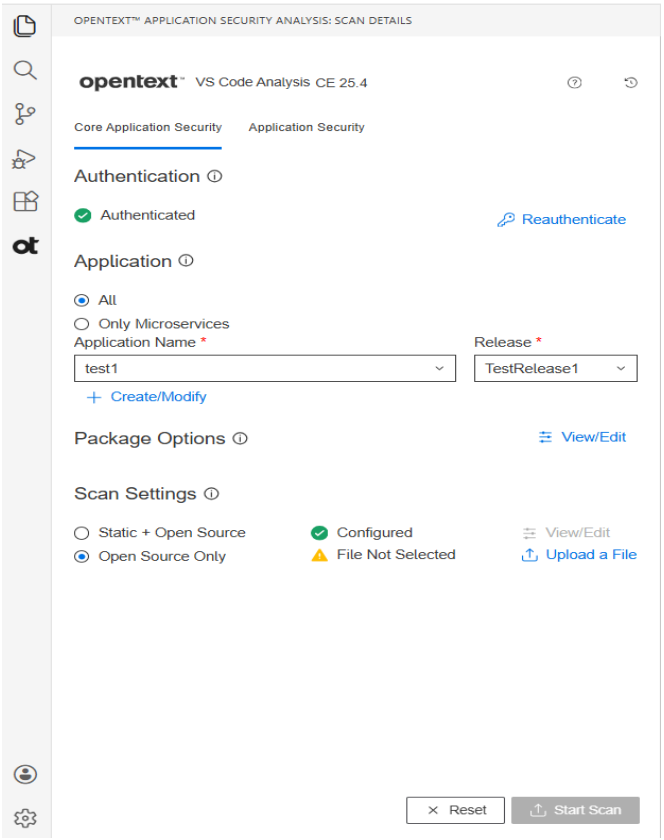
You must have the standalone ScanCentral SAST client on the system where OpenText™ Application Security Extensions for Visual Studio Code is installed to upload code to OpenText™ Core Application Security. You can obtain the ScanCentral SAST client from the OpenText™ Core Application Security Tools page. For instructions on how to install the ScanCentral SAST client, see the README file included in the ZIP archive.

To upload the opened project to OpenText Core Application Security for assessment:

Authenticate your OpenText Core Application Security account

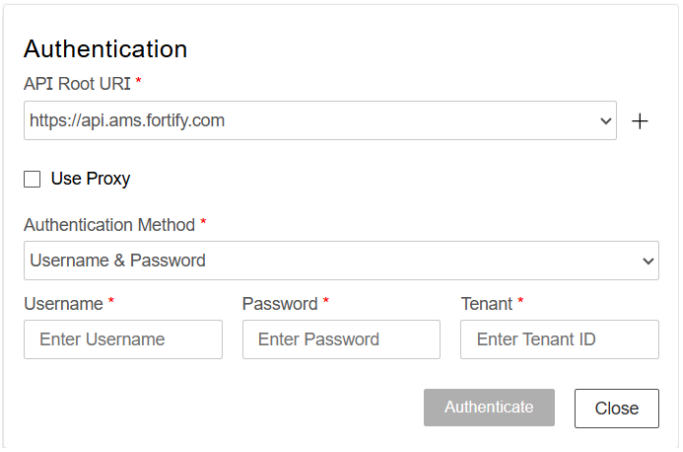
1. If the extension is not open, click **OpenText™ Application Security Analysis** in the activity bar.

   By default, the **Core Application Security** tab displays.



2. In the **Authentication** area, click **Authenticate**.

   The Authentication box is displayed in the VS Code Editor.



3. From the **API Root URI** list, you can either select one of the available the API root URL.

   Alternatively, if your API root URL is not available:

   1. Click + to add a new API root URL to the **API Root URI** list.
   2. Type a valid API root URL.
   3. Click ✓. If the API root URL is valid, you can view the new URL in the **API Root URL** list.

4. (Optional) Select the **Use Proxy** check box to connect through a proxy and provide the settings described in the following table.

Field	Description
Host	Type the name of the proxy server. Exclude the protocol from the proxy host (for example, some.secureproxy.com).
Port	Type the port of the proxy server.
Proxy credentials	Select the check box if the proxy server requires authentication. Type the account credentials for the proxy server.
Use HTTPS	Turn on the **Use HTTPS** switch to connect using HTTPS.

5. Select an authentication method and provide the relevant credentials described in the following table.

Authentication method	Procedure
Username & Password	1. In the **Username** box, type the account username. 2. In the **Password** box, type the account password. 3. In the **Tenant** box, type the tenant ID.
Personal Access Token	1. In the **Username** box, type the account username. 2. In the **Access Token** box, type the personal access token. 3. In the **Tenant** box, type the tenant ID.
Authentication Token	1. In the **Client Id** box, type the API key. 2. In the **Client Secret** box, type the API secret.

6. Click **Authenticate**.

   Upon successful authentication, **Authenticated** status is displayed. Otherwise, the status displays **Not Authenticated**.

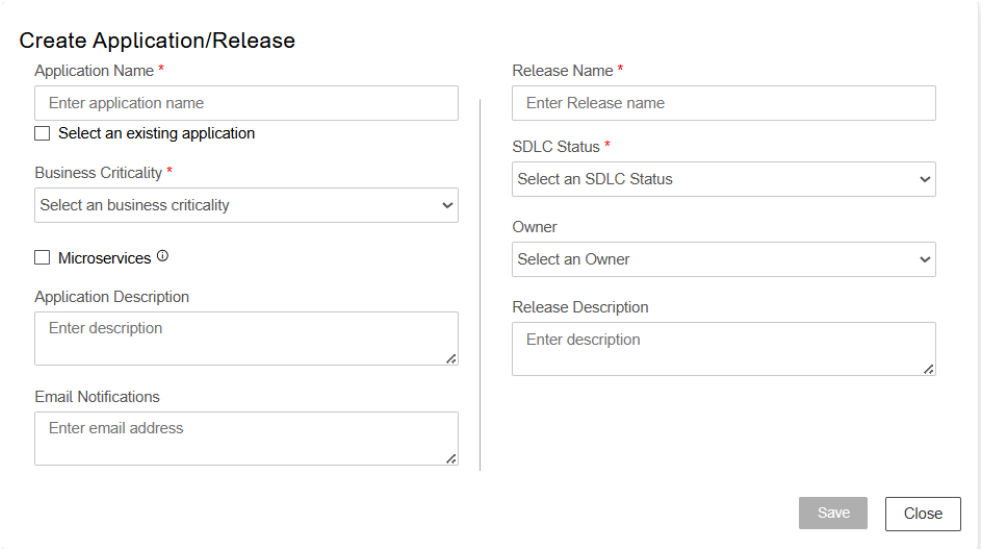
   To modify the API root URL, authentication method, or credentials, click **Reauthenticate**.

Select Application and Release

1. In the **Application** area, select **All** or **Microservices**.
2. Select an application or a microservice from the **Application Name** list.

3. Select a release from the **Release** list.
4. To create a new Application or Release or modify an existing application or release, click **Create/Modify**.

   The **Create Application/Release** box displays in the VS Code editor.



5. In the **Create Application/Release** box, define the application. Fields are required, unless otherwise noted.

Property	Description
Application Name	Type the name of your application or use the **Select an existing application** check box to modify an existing application.
Select an existing application	To modify an existing application, select the check box. Select or search for an application from the **Application Name** list.
Business Criticality	Select the application's level of importance: ◦ **High**: Security issues could have catastrophic consequences for the business. ◦ **Medium**: Security issues would have non-trivial consequences, but ones which do not pose a life-or-death threat to the business. ◦ **Low**: Security issues can be ignored or addressed gradually as time permits
Microservices	(**Web / Thick-Client** applications only) Select the check box to scan the application as a microservice application. **Important** The designation of a microservice application is permanent and cannot be changed after the application has been created.

Application Description	(Optional) Type a description of the application that will help you manage multiple applications.
Email Notifications	(Optional) List the email addresses that will receive email notifications of scan status updates for the application. Separate multiple email addresses with a semicolon or comma.

6. After you specify the application details, you can define the release. Fields are enabled after you specify the application details. Fields are required, unless otherwise noted.

Property	Description
Release Name	Type the name of your release or use the **Select an existing Release** check box to modify an existing release.
Selecting an existing Release	To modify an existing release, select the check box. Select or search for a release from the **Release Name** list.
SDLC Status	Select the Software Development Life Cycle stage of the release: **Development**, **QA/Test**, **Production**. The **Retired** option is not available.
Owner	Owner of the release who receives email notifications of scan status updates to the release
Release Description	(Optional) Type a description that helps describe the release.

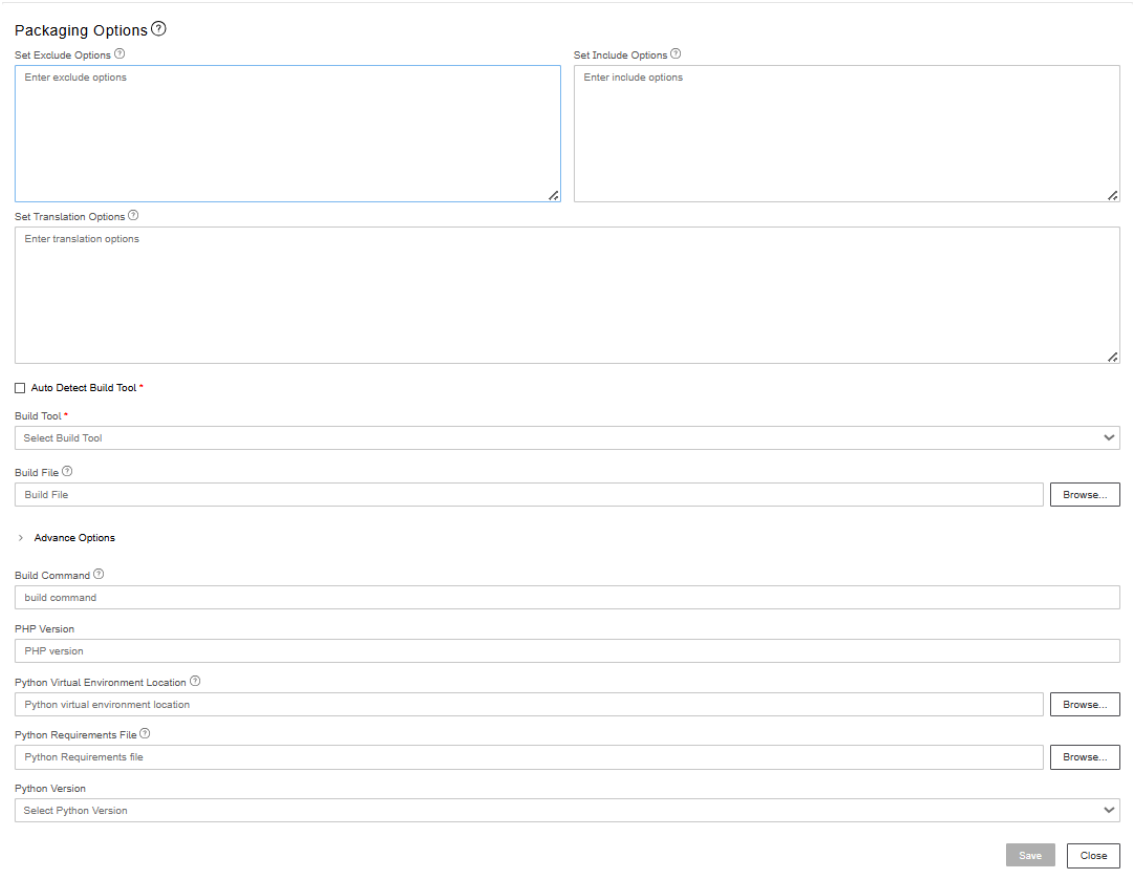
Note

You can add and modify custom attributes for application, releases, or microservices that are marked Required for the OpenText Core Application Security tenant.

ScanCentral Package Options

1. In the **Package Options** area, click **View/Edit**.

   The Packaging Options box is displayed.



2. Provide the options described in the following table. All field are optional, unless otherwise noted.

Option	Description
Set Exclude Options	Specify the relative paths of files or directories to exclude from the package separated by a new line (one per line). You can use wildcard characters to specify the paths.
Set Include Options	Specify the relative paths of files or directories to include in the package separated by a new line (one per line). You can use wildcard characters to specify the paths. If you leave it empty, all the supported files are included.
Set Translation Options	Specify a list of OpenText SAST translation options separated by a new line (one per line).

Option	Description
Auto Detect Build Tool	Select this option to automatically detect the build tool. **Note** On Windows agents, if you select the **Auto Detect build tool** option, the default build tool is MSBuild. On Linux agents, if you select the **Auto Detect build tool** option, the default build tool is DotNet. If you want to use DotNet as the build tool on a Windows agent, you must clear the **Auto Detect build tool** option and explicitly select **DotNet** from the **Build Tool** list.
Advanced Options	
Skip Build	Select whether to skip the build invocation that prepares the generated sources and libraries before the project information is packaged for submission to ScanCentral SAST.

Option	Description
Set Debug	Select this option to generate a ZIP file that includes debug log files from clients, sensors, and OpenText SAST.

3. To select a build tool explicitly, select the build in the **Build Tool** list and provide the settings based on the build tool.
    1. If you selected **DotNet**, **Gradle**, **Maven**, or **MSBuild** in the **Build Tool** list, provide the information described in the following table.

Option	Description
Build File	(For DotNet, Gradle, or Maven) Click **Browse** and select the build file if it is different than the default of `build.gradle` or `pom.xml`. (For MSBuild) Type the name of the build file. If you do not select a build file, ScanCentral SAST automatically detects the build file.
Set Exclude Options	Specify the relative paths of files or directories to exclude from the package separated by a new line (one per line). You can use wildcard characters to specify the paths.
Set Include Options	Specify the relative paths of files or directories to include in the package separated by a new line (one per line). You can use wildcard characters to specify the paths. If you leave it empty, all the supported files are included.
Set Translation Options	Specify a list of OpenText SAST translation options separated by a new line (one per line).
Build Command	(Optional) Type any custom build commands to prepare and build the project. If not specified, the default build command is used.

Option	Description
Advanced Options	
Skip Build	Select whether to skip the build invocation that prepares the generated sources and libraries before the project information is packaged for submission to Fortify ScanCentral SAST. **Note** This setting is only valid for Gradle and Maven.
Set Debug	Select this option to generate a ZIP file that includes debug log files from clients, sensors, and OpenText SAST.

2. If you selected **None** in the **Build Tool** list, provide the information described in the following table.

Option	Description
PHP Version	(Optional) Type the PHP version used in the project.
Python Virtual Environment Location	(Optional) Click **Browse** and select the location (directory) of the Python virtual environment. Specify this together with the Python requirements file to have dependencies restored before the scan.
Python Requirements File	(Optional) Click **Browse** and select the Python project requirements file used to install and collect dependencies. Use only this Python field if you have no preference for the Python version used or there is only one Python version installed and on the PATH.
Python Version	(Optional) Select the Python version for Python projects.

4. Click **Save**.

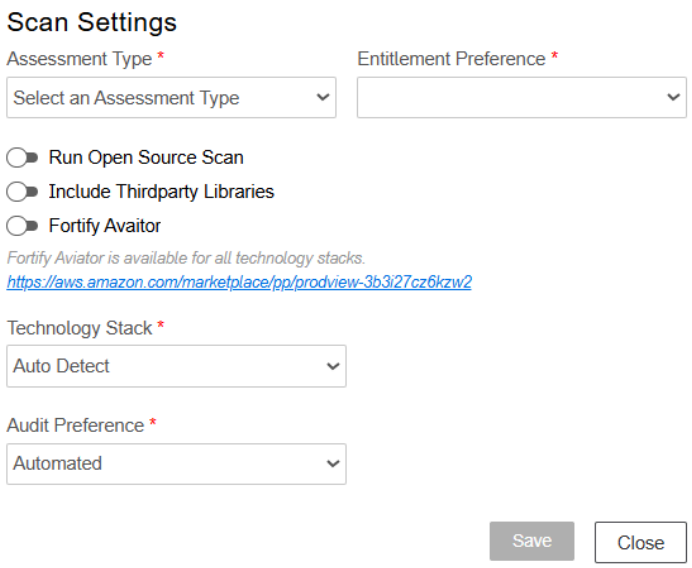
Configure Scan Settings

In the **Scan Settings** area, you can select either a Static assessment or an open source software composition analysis in conjunction with a static assessment or as a separate open source only assessment.

Configure a Static Scan

1. In the **Scan Settings** area, select **Static + Open Source.**
2. If the status displays **Not Configured**, click **View/Edit**.

   The Scan Settings box displays.



3. Complete the fields as needed. All field are required, unless otherwise noted.

Field	Description
Assessment Type	Select **Static+ Assessment** or **Static Assessment**.
Entitlement Preference	Select the entitlement that the assessment will use. The field displays entitlements that are valid for the selected assessment type, including those available for purchase. Note that microservice applications are restricted to subscriptions. If the release has an active subscription, only options that do not consume entitlements are displayed.
Run Open Source Scan	(Optional) Select the check box to include open source software composition analysis. No code leaves the OpenText Core Application Security environment.
Include Third Party Libraries	(Optional) Select the check box to have third party libraries scanned for vulnerabilities, which will be included in the scan results. This significantly increases the turnaround time. This option is not available for microservice applications.  **Note** Selecting this option infers that your organization has received consent from all third-party vendors to scan their libraries.

Field	Description
SAST Aviator	(Optional) For scans using Automated audit, select the check box to have SAST Aviator audit results and provide enhanced remediation assistance.
Technology Stack	Select the application's technology stack. The languages available for selection depends on the application type and whether the application is a microservice application.
Language Level	If applicable, select the technology stack's language level from the list.

Field	Description
Audit Preference	Select the audit preference. ◦ **Manual**: False positives identified by Fortify Audit Assistant with high confidence are automatically suppressed. A security expert then manually reviews the scan results. ◦ **Automated**: False positives identified by Fortify Audit Assistant with high confidence are automatically suppressed and results are published without manual review. **Note** If you select the SAST Aviator check box, Automated audit is selected by default. If you change the **Audit Preference** to **Manual**, the **SAST Aviator** check box is cleared automatically. The ability to select audit preference depends on the assessment type: ◦ A Static single scan allows Automated only. ◦ A Static subscription allows one Manual audit per application (not per release or microservice). ◦ A Static+ single scan allows Manual only. ◦ A Static+ subscription allows Automated or Manual audit for each assessment.

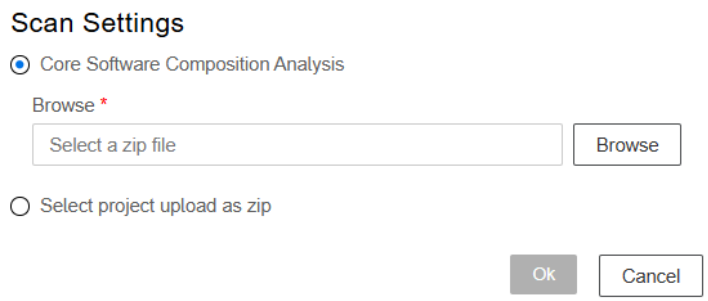
4. Click **Save**.
5. Click **Start Scan** to upload a ScanCentral package to OpenText Core Application Security.

   If the project is successfully uploaded, the OpenText™ Application Security Extensions for Visual Studio Code log displays the completion status and the scan ID. The OpenText Core Application Security Scans pages display a new scan for the release.
6. Click (**Recent Scans**). The **Recent Scans** box displays the recent 10 scans from OpenText Core Application Security and details such as Scan ID, application, release, scan type, status, start time, and completion time. If you want to cancel an in-progress scan, click **Cancel**.

Configure an Open Source Only Scan

1. In the **Scan Settings** area, Select **Open Source Only**.
2. If the status displays **File Not Selected**, click **Upload a File**.

   The Scan Settings box displays.



3. Complete the fields as needed.

Field	Description
Core Software Composition Analysis	Selected by default. Run a Debricked open source scan to generate a Software Bill of Materials. Click **Browse** and navigate to and select your zip file containing the lock file. The zip file can contain either the static assessment payload or just the lock file.
Select project upload as zip	Select this option to package the current project in the workspace and upload the project as a zip file.

4. Click **Ok.**

   The status displays the zip file name or the Current Project.

5. Click **Start Scan** to upload the package to OpenText Core Application Security.

   If the project is successfully uploaded, the OpenText™ Application Security Extensions for Visual Studio Code log displays the completion status and the scan ID. The OpenText Core Application Security **Scans** page displays a new scan for the release.

6. Click (**Recent Scans**). The **Recent Scans** box displays the recent 10 scans from OpenText Core Application Security and details such as Scan ID, queued time, application, release, scan type, status, start time, and completion time. If you want to cancel an in-progress scan, click **Cancel**.

The application, release, and scan settings are available when you perform a quick scan for the current project. For more information, see Performing a quick scan.

The scan settings are saved when you log out and automatically retrieved when you re-authenticate your OpenText Core Application Security credentials.

If you want to reset the saved options, click **Reset**.

1.3.2. Performing a local analysis with OpenText SAST

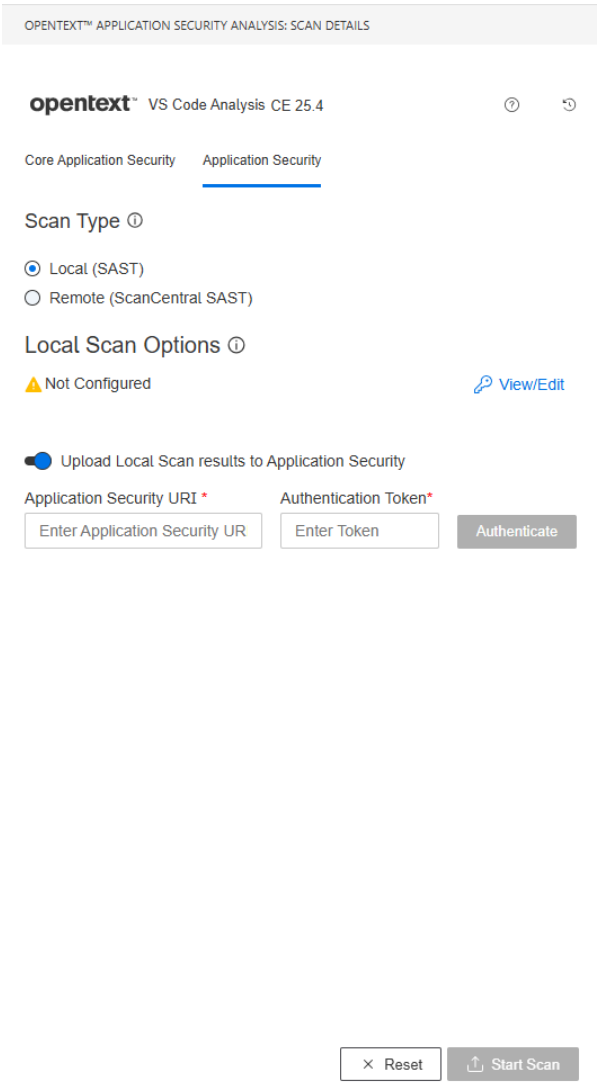
Requirements

- You must have OpenText SAST locally installed.
- Ensure that the workspace completely loads in the Visual Studio Code before you start the extension.

Configuring a local analysis with OpenText SAST

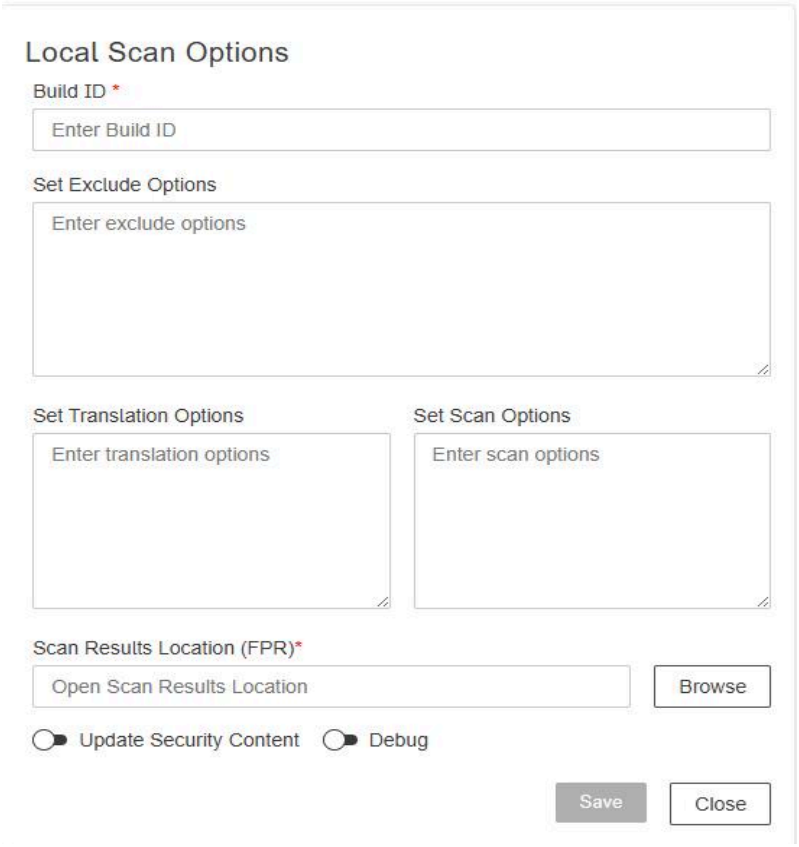
To scan the opened project locally with OpenText SAST:

1. If the extension is not open, click **OpenText™ Application Security Analysis** in the activity bar.
2. Click the **Application Security** tab.



3. In the **Scan Type** area, select **Local (SAST)**.

4. In the **Local Scan Options** area, click **View/Edit**.



5. Provide the information described in the following table.

Option	Description
Build ID	Required. Type a unique identifier for the analysis.
Set Exclude Options	Optional. Specify the relative paths of files or directories to exclude from the package separated by a new line (one per line). You can use wildcard characters to specify the paths.
Set Translation Options	Optional. Specify a list of OpenText SAST translation options separated by a new line (one per line).
Set Scan Options	Optional. Specify a list of OpenText SAST scan options separated by a new line (one per line).
Scan Results Location (FPR)	Required. Click **Browse** and select the Fortify Project Results file (for example, `MyProjectA.fpr`). If you do not provide an analysis results file name, then OpenText™ Application Security Extensions for Visual Studio Code uses the name of the current project folder for the FPR file and saves the FPR in the current project folder.
Update Security Content	Optional. Enable this option to download Application Security content before the scan.
Debug	Optional. Select this option to display debug information that can be helpful to troubleshoot issues.

6. Click **Save**.

   The scan options are saved locally in a `WorkspaceSettings.json` file located in the `.otappsec/<version>` directory of the project in the current workspace.

7. Click **Start Scan**.

   OpenText™ Application Security Analysis Extension for Visual Studio Code automatically detects the OpenText SAST path from the Visual Studio Code extension settings.

   OpenText™ Application Security Analysis starts the scan and displays the status information. When the scan is complete, OpenText™ Application Security Analysis Extension for Visual Studio Code displays the scan completion status in an information message.

8. Click (**Recent Scans**). The **Recent Scans** box displays the recent 10 scans from Application Security and details such as Scan ID, application, version, scan type, status, start time, and completion time. If you want to cancel an in-progress scan, click **Cancel**.

The analysis process includes the following phases:

- During the clean phase, OpenText SAST removes files from a previous translation of the project.
- During the translation phase, you can see one translation section for each of the selected modules. OpenText SAST translates source code identified in the previous page into an intermediate format associated with the build ID. (The build ID is typically the project name.)
- During the scan phase, OpenText SAST analyzes the source files identified during the translation phase and generates analysis results in the FPR format.

(Optional) Uploading local scan results to Application Security

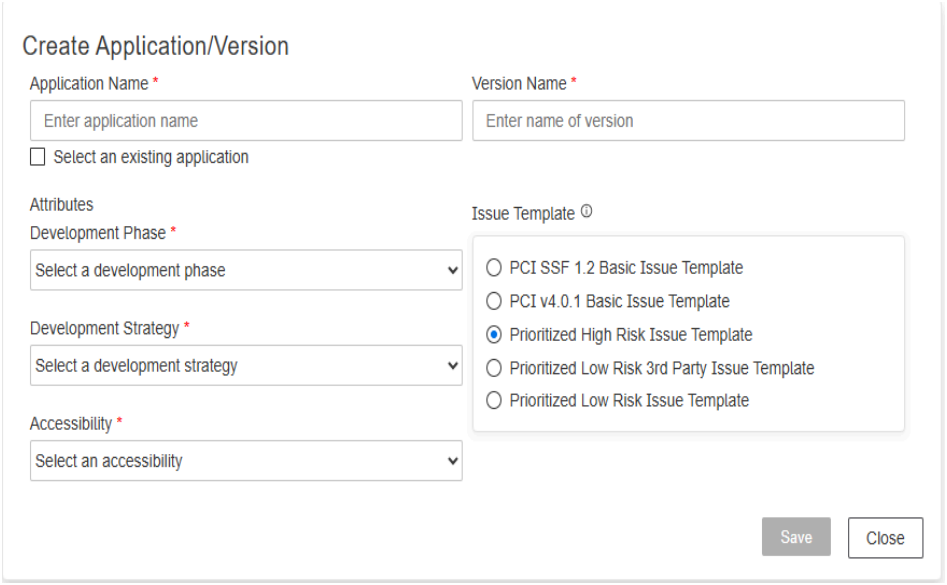
1. To upload results to Application Security, turn on the **Upload Local Scan results to Application Security** switch.

   Before uploading results to Application Security, you need to authenticate your Application Security account.

Field	Description
Application Security URI	Type the server URL to connect to Application Security.
Authentication Token	Specify the encoded value of an Application Security authentication token of type `ToolsConnectToken`.

2. Click **Authenticate**. The **Application Name** list and **Version** list is displayed.
3. Select the application name in the **Application Name** list and the version name in the **Version** list.
4. Alternatively, click **Create New** to create a new application or version or modify an existing application or version.

   The Create Application/Version box is displayed.



5. Provide the information described in the following table.

Field	Description
Application Setup	
Application Name	(Required) Type the application name or use the **Select an existing application** check box to modify an existing application.
Select an existing application	To modify an existing application, select the check box. Select or search for an application from the **Application Name** list.
Version Setup	
Version Name	(Required) Type a name of the version or use the **Select an existing version** check box to modify an existing version.
Select an existing version	To modify an existing version, select the check box. Select or search for a version from the **Version Name** list.
Attributes	
Development Phase	(Required) Current phase of development the application version is in.
Development Strategy	(Required) Staffing strategy used for application development.
Accessibility	(Required) Level of access required to use the application.

Field	Description
Issue Template	Select the check box for a template that sets the minimum thresholds for issue detection. The default template is Prioritized High Risk Issue Template.

6. Click **Save**.
7. Click **Start Scan**.

   OpenText™ Application Security Analysis starts the scan and displays the status information. When the scan is complete, OpenText™ Application Security Analysis Extension for Visual Studio Code uploads the FPR file to Application Security.

The scan settings are available when you perform a quick scan for the current project. For more information, see Performing a quick scan.

If you want to reset the saved scan settings, click **Reset**.

1.3.3. Performing an analysis remotely with ScanCentral SAST

This section describes the requirements, configuration, and procedure to use ScanCentral SAST to analyze your code.

With the OpenText™ Application Security Analysis Extension for Visual Studio Code and ScanCentral SAST, you can either:

- Perform the entire analysis (translation and scan) remotely with ScanCentral SAST
- Perform the translation locally with OpenText SAST and then automatically upload the translated project to ScanCentral SAST for the scan phase.

  You must translate the project locally if it uses a language that ScanCentral SAST does not support in remote translation. For a list of supported languages, see the *Fortify Software System Requirements* document.

  You must have a locally installed and licensed OpenText SAST to perform the translation phase.

  Make sure that the Application Security Content version on the local system is the same as the version on the Fortify ScanCentral sensor. OpenText strongly recommends that you periodically update the security content.

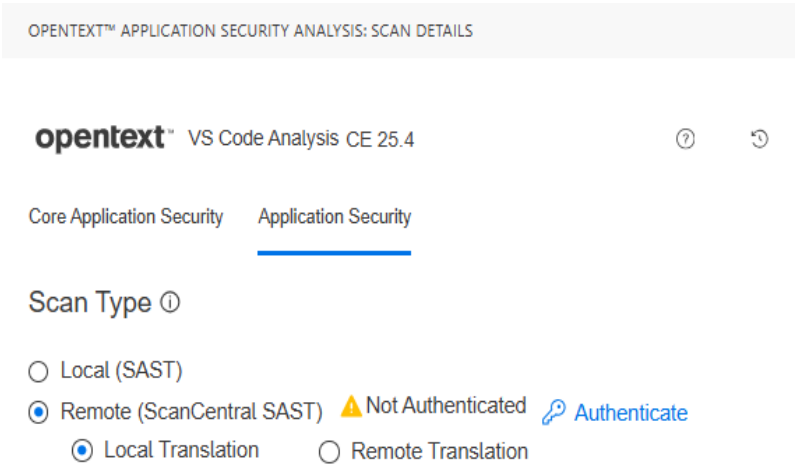
Requirements

- You must have a properly configured ScanCentral SAST installation. For more information, see the *OpenText™ ScanCentral SAST Installation, Configuration, and Usage Guide* in OpenText Application Security Documentation.
- An Application Security URL and an authentication token of type ToolsConnectToken for a server that is integrated with ScanCentral SAST.
- A locally installed and licensed OpenText SAST with Application Security Content.

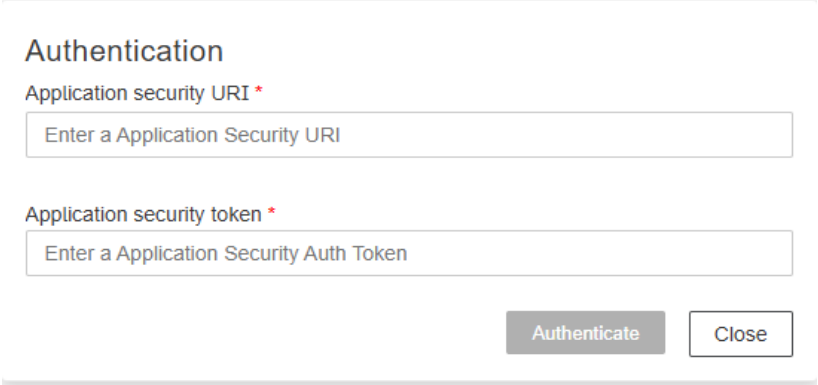
To upload the opened project for analysis by ScanCentral SAST:

Authenticate your OpenText™ Application Security account:

1. If the extension is not open, click **OpenText™ Application Security Analysis** in the activity bar.
2. Click the **Application Security** tab.
3. In the **Scan Type** area, select **Remote (ScanCentral SAST)**.



4. Click **Authenticate**. The Authentication box is displayed in the VS Code editor.



5. Provide the information described in the following table.

Field	Description
Application Security URI	Type the server URL to connect to Application Security.
Application Security Token	Specify the encoded value of an Application Security authentication token of type `ToolsConnectToken`.

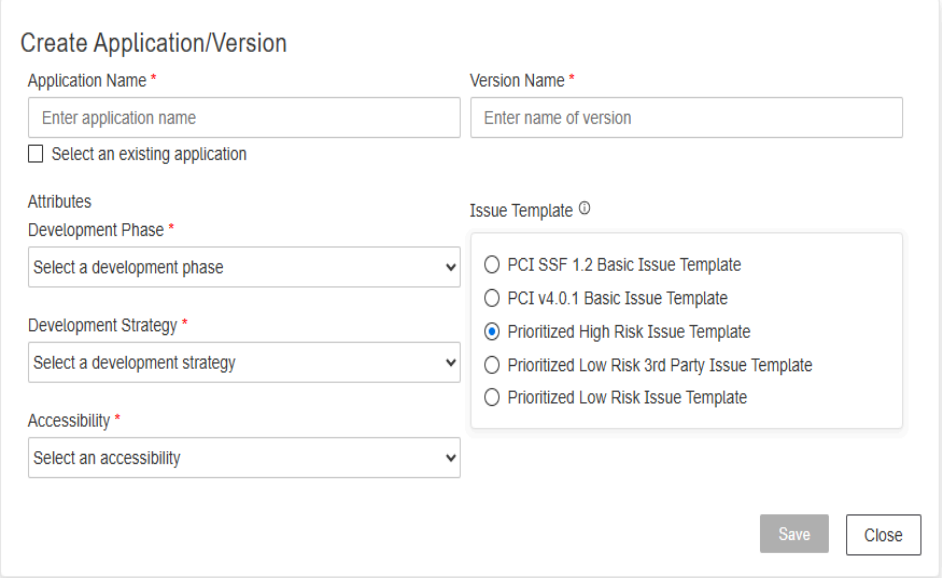
6. Click **Authenticate**.

Configuring a remote scan with local translation

1. Select **Remote (ScanCentral SAST) > Local Translation** to run the translation phase on the local system and the scan phase with ScanCentral SAST.
2. In the **Application** area, select an existing application or create a new application.

1. Select the application name in the **Application Name** list and the version name in the **Version** list.
2. To create a new Application or Version or modify an existing application or version, click **Create New** to create a new application or version or modify an existing application or version.

   The Create Application/Version box is displayed.



3. Provide the information described in the following table.

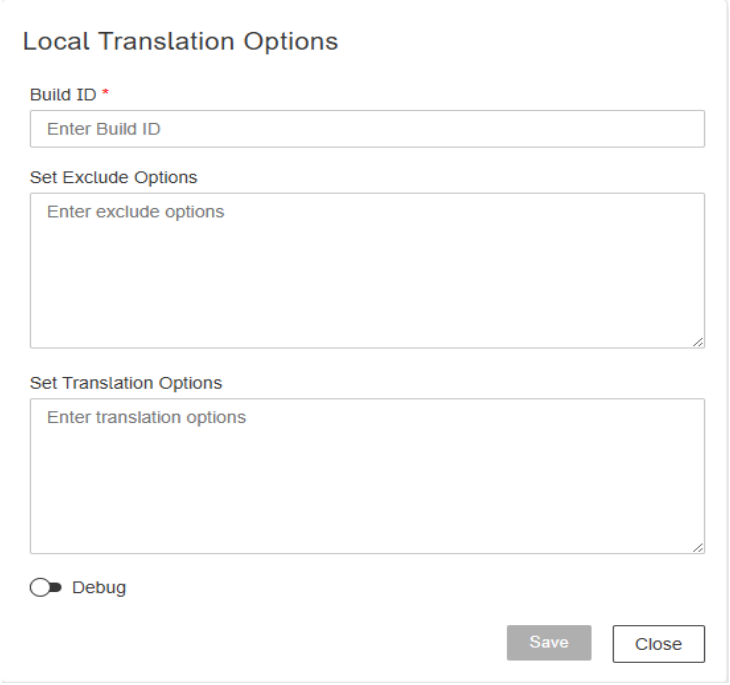
Field	Description
Application Setup	
Application Name	(Required) Type the application name or use the **Select an existing application** check box to modify an existing application.
Select an existing application	To modify an existing application, select the check box. Select or search for an application from the **Application Name** list.
Version Setup	
Version Name	(Required) Type a name of the version or use the **Select an existing version** check box to modify an existing version.
Select an existing version	To modify an existing version, select the check box. Select or search for a version from the **Version Name** list.
Attributes	
Development Phase	(Required) Current phase of development the application version is in.
Development Strategy	(Required) Staffing strategy used for application development.
Accessibility	(Required) Level of access required to use the application.

Field	Description
Issue Template	Select the check box for a template that sets the minimum thresholds for issue detection. The default template is Prioritized High Risk Issue Template.

4. Click **Save**.

3. In the **Local Translation Options** area, click **View/Edit**.

   The **ScanCentral Local Translation Options** box is displayed in the VS Code editor.

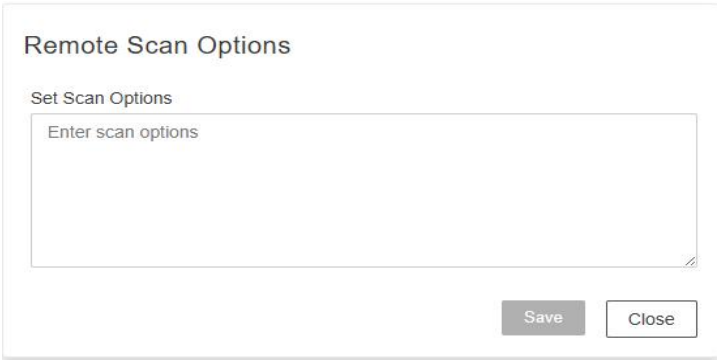


4. Complete the fields as needed.

Option	Description
Build ID	Type a unique identifier for the analysis.
Set Exclude Options	Specify the relative paths of files or directories to exclude from the package separated by a new line (one per line). You can use wildcard characters to specify the paths.
Set Translation Options	Specify a list of OpenText SAST translation options separated by a new line (one per line).
Set Translation Options	Specify a list of OpenText SAST translation options separated by a new line (one per line).
Debug	Select this option to display debug information that can be helpful to troubleshoot issues.

5. Click **Save**.
6. (Optional) In the **Remote Scan Options** area, click **View/Edit**.

   The **ScanCentral Remote Scan Options** box is displayed in the VS Code editor.



7. In the **Set Scan Options** field, specify a list of ScanCentral SAST scan options separated by a new line (one per line).
8. Click **Save**.

> **Note**
> The scan type, application, version, translation options, and remote scan options are saved locally in a `WorkspaceSettings.json` file located in the `.otappsec/<version>` directory of the project in the current workspace.

9. Click **Start Scan**.

   OpenText™ Application Security Analysis Extension for Visual Studio Code automatically detects the file path to ScanCentral SAST client executable and the locally installed OpenText SAST executable from the Visual Studio Code extension settings.

   OpenText™ Application Security Analysis starts the scan and displays the status information. When the scan is complete, OpenText™ Application Security Analysis Extension for Visual Studio Code displays the scan completion status in an information message.

10. Click (**Recent Scans**). The **Recent Scans** box displays the recent 10 scans from Application Security and details such as Scan ID, application, version, scan type, status. If you want to cancel an in-progress scan, click **Cancel**.

OpenText™ Application Security Analysis Extension for Visual Studio Code scans the code in the following phases:

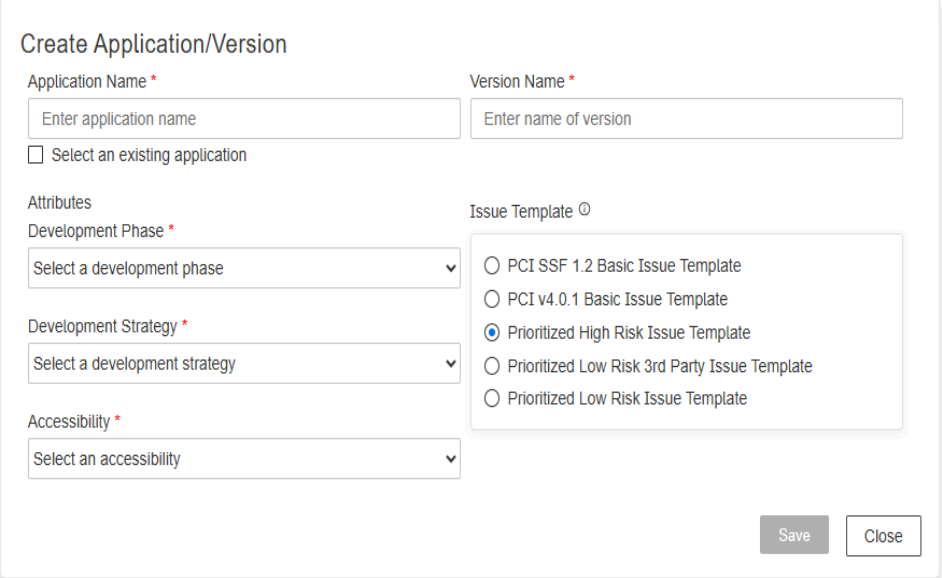
- During the *clean* phase, OpenText SAST removes files from a previous translation of the project.
- During the *translation* phase, you can see one translation section for each selected module. You can change the class path and build parameters for each module individually. OpenText SAST translates source code identified in the previous page into an intermediate format associated with the build ID. (The build ID is typically the project name.)

  Any additional OpenText SAST translation options configured on the **Advanced Options** tab in the analysis settings are shown here. You can change any of the OpenText SAST options. For information about the available command-line options, see the *OpenText™ Static Application Security Testing User Guide.*
- During the scan phase, ScanCentral SAST analyzes the source files identified during the translation phase and generates analysis results in the FPR format.

Configuring a remote translation and scan

1. Select **Remote (ScanCentral SAST) > Remote Translation** to run the translation phase and the scan phase with ScanCentral SAST.
2. In the **Application** area, select an existing application or create a new application.
    1. Select the application name in the **Application Name** list and the version name in the **Version** list.
    2. To create a new Application or Version or modify an existing application or version, click **Create New** to create a new application or version or modify an existing application or version.

       The Create Application/Version box is displayed.



    3. Provide the information described in the following table.

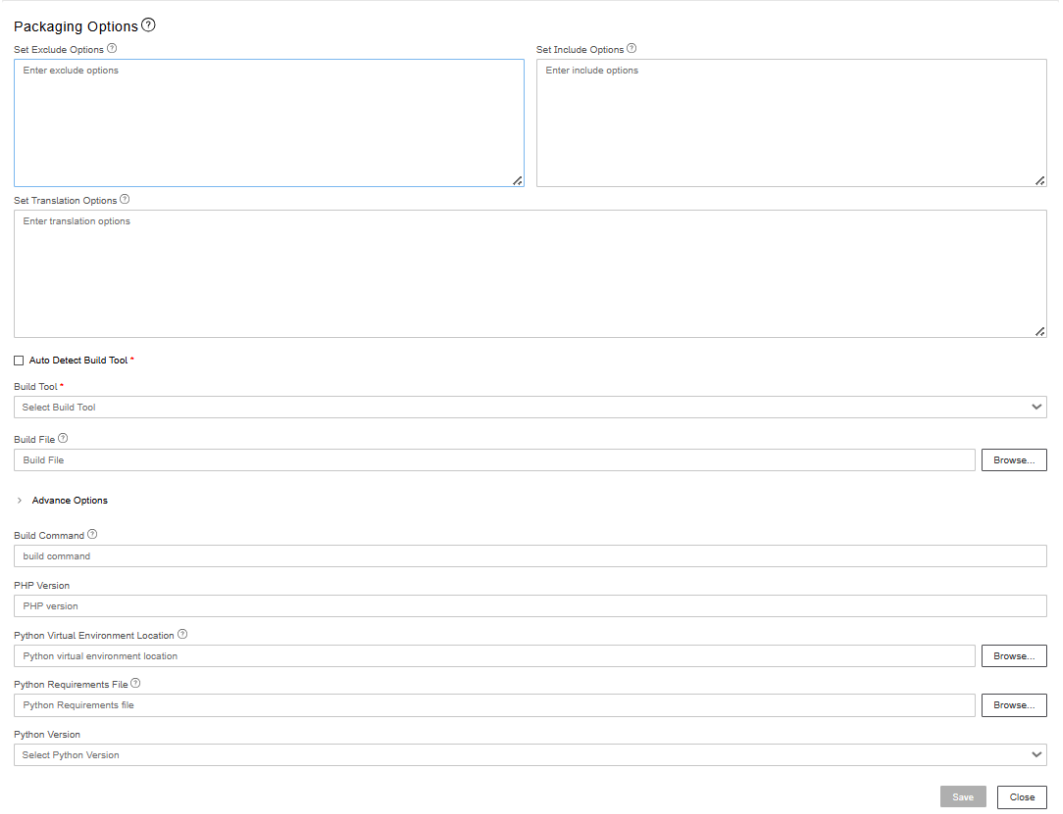
Field	Description
Application Setup	
Application Name	(Required) Type the application name or use the **Select an existing application** check box to modify an existing application.
Select an existing application	To modify an existing application, select the check box. Select or search for an application from the **Application Name** list.
Version Setup	
Version Name	(Required) Type a name of the version or use the **Select an existing version** check box to modify an existing version.
Select an existing version	To modify an existing version, select the check box. Select or search for a version from the **Version Name** list.
Attributes	
Development Phase	(Required) Current phase of development the application version is in.
Development Strategy	(Required) Staffing strategy used for application development.
Accessibility	(Required) Level of access required to use the application.

Field	Description
Issue Template	Select the check box for a template that sets the minimum thresholds for issue detection. The default template is Prioritized High Risk Issue Template.

4. Click **Save**.

3. In the **Package Options** area, click **View/Edit**.

   The Packaging Options box is displayed in the VS Code editor.



4. Provide the options described in the following table. All field are optional, unless otherwise noted.

Option	Description
Set Exclude Options	Specify the relative paths of files or directories to exclude from the package separated by a new line (one per line). You can use wildcard characters to specify the paths.
Set Include Options	Specify the relative paths of files or directories to include in the package separated by a new line (one per line). You can use wildcard characters to specify the paths. If you leave it empty, all the supported files are included.
Set Translation Options	Specify a list of OpenText SAST translation options separated by a new line (one per line).
Auto Detect Build Tool	Select this option to automatically detect the build tool. **Note** On Windows agents, if you select the **Auto Detect Build Tool** option, the default build tool is MSBuild. On Linux agents, if you select the **Auto Detect Build Tool** option, the default build tool is DotNet. If you want to use DotNet as the build tool on a Windows agent, you must clear the **Auto Detect Build Tool** option and explicitly select **DotNet** from the **Build Tool** list.

Option	Description
Advanced Options	
Skip Build	Select whether to skip the build invocation that prepares the generated sources and libraries before the project information is packaged for submission to ScanCentral SAST.
Set Debug	Select this option to generate a ZIP file that includes debug log files from clients, sensors, and OpenText SAST.

5. To select a build tool explicitly, clear the **Auto Detect build tool** option. The **Build Tool** list is displayed.
6. In the **Build Tool** list, select the build and provide the settings based on the build tool.
    1. If you selected **DotNet**, **Gradle**, **Maven**, or **MSBuild** in the **Build Tool** list, provide the information described in the following table.

Option	Description
Build File	(For DotNet, Gradle, or Maven) Click **Browse** and select the build file if it is different than the default of `build.gradle` or `pom.xml`. (For MSBuild) Type the name of the build file. If you do not select a build file, ScanCentral SAST automatically detects the build file.
Set Exclude Options	Specify the relative paths of files or directories to exclude from the package separated by a new line (one per line). You can use wildcard characters to specify the paths.
Set Include Options	Specify the relative paths of files or directories to include in the package separated by a new line (one per line). You can use wildcard characters to specify the paths. If you leave it empty, all the supported files are included.
Set Translation Options	Specify a list of OpenText SAST translation options separated by a new line (one per line).
Build Command	(Optional) Type any custom build commands to prepare and build the project. If not specified, the default build command is used.

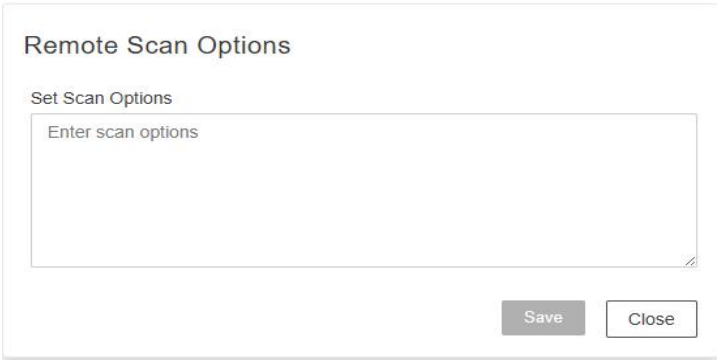
Option	Description
Advanced Options	
Skip Build	Select whether to skip the build invocation that prepares the generated sources and libraries before the project information is packaged for submission to Fortify ScanCentral SAST. This setting is only valid for Gradle and Maven.
Set Debug	Select this option to generate a ZIP file that includes debug log files from clients, sensors, and OpenText SAST.

2. If you selected **None** in the **Build Tool** list, provide the information described in the following table.

Option	Description
PHP Version	(Optional) Type the PHP version used in the project.
Python Virtual Environment Location	(Optional) Click **Browse** and select the location (directory) of the Python virtual environment. Specify this together with the Python requirements file to have dependencies restored before the scan.
Python Requirements File	(Optional) Click **Browse** and select the Python project requirements file used to install and collect dependencies. Use only this Python field if you have no preference for the Python version used or there is only one Python version installed and on the PATH.
Python Version	(Optional) Select the Python version for Python projects.

7. Click **Save**.
8. (Optional) In the **Remote Scan Options** area, click **View/Edit**.

   The **ScanCentral Remote Scan Options** box is displayed in the VS Code editor.



9. In the **Set Scan Options** field, specify a list of ScanCentral SAST scan options separated by a new line (one per line).

10. Click **Save**.

> **Note**
> The scan type, application, version, translation options, and remote scan options are saved locally in a `WorkspaceSettings.json` file located in the `.otappsec/<version>` directory of the project in the current workspace.

11. Click **Start Scan**.

    OpenText™ Application Security Analysis Extension for Visual Studio Code automatically detects the file path to ScanCentral SAST client executable from the Visual Studio Code extension settings.

    OpenText™ Application Security Analysis starts the scan and displays the status information. When the scan is complete, OpenText™ Application Security Analysis Extension for Visual Studio Code displays the scan completion status in an information message.

12. Click (**Recent Scans**). The **Recent Scans** box displays the recent 10 scans from Application Security and details such as Scan ID, application, version, scan type, status, start time, and completion time. If you want to cancel an in-progress scan, click **Cancel**.

OpenText™ Application Security Analysis Extension for Visual Studio Code scans the code in the following phases:

- During the *clean* phase, OpenText SAST removes files from a previous translation of the project.
- During the *translation* phase, you can see one translation section for each selected module. You can change the class path and build parameters for each module individually. OpenText SAST translates source code identified in the previous page into an intermediate format associated with the build ID. (The build ID is typically the project name.)

  Any additional OpenText SAST translation options configured on the **Advanced Options** tab in the analysis settings are shown here. You can change any of the OpenText SAST options. For information about the available command-line options, see the *OpenText™ Static Application Security Testing User Guide.*
- During the scan phase, ScanCentral SAST analyzes the source files identified during the translation phase and generates analysis results in the FPR format.

You can view the analysis results on Application Security if you uploaded them to the server.

The scan settings are available when you perform a quick scan for the current project. For more information, see Performing a quick scan.

The scan settings and options are saved when you close the VS Code and automatically retrieved when you authenticate your OpenText Application Security credentials.

If you want to reset the saved options to default, click **Reset**.

1.3.4. Performing a quick scan

Quick scan provides a way to quickly package and scan your currently opened project using the recently saved configurations.

Requirements

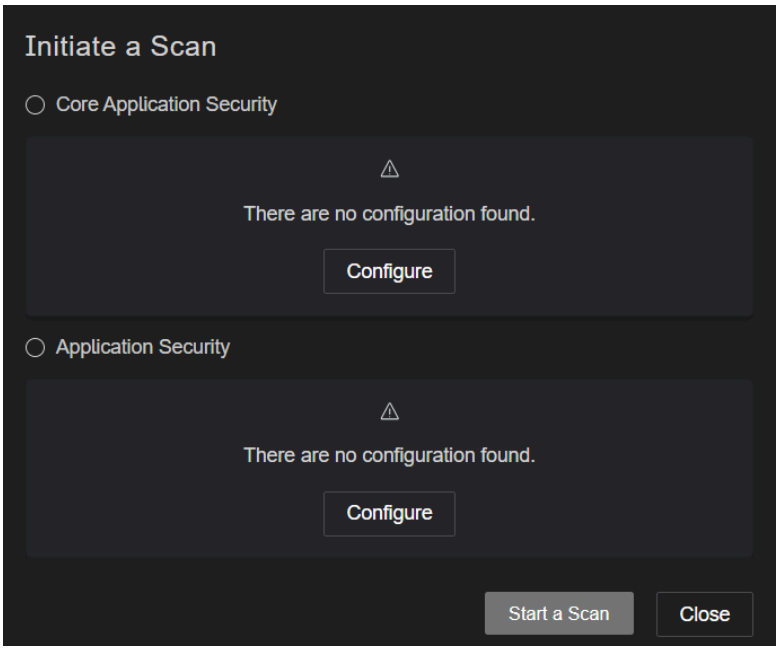
Before you start a scan:

- Ensure you have configured the VS Code Settings for OpenText Application Security Analysis. See Configuring the Application Security Analysis Extension for Visual Studio Code.
- If you want to scan and upload the project to OpenText Core Application Security, ensure you have configure the scan settings for Core Application Security. See Uploading Code to OpenText Core Application Security for Assessment.
- If you want to scan locally and upload the project to Application Security, ensure you have configure the local scan settings for Application Security. See Performing a local analysis with OpenText SAST.
- If you want to perform a remote analysis, ensure you have configure the remote scan settings for Application Security. See Performing an analysis remotely with ScanCentral SAST.

Initiate a Scan

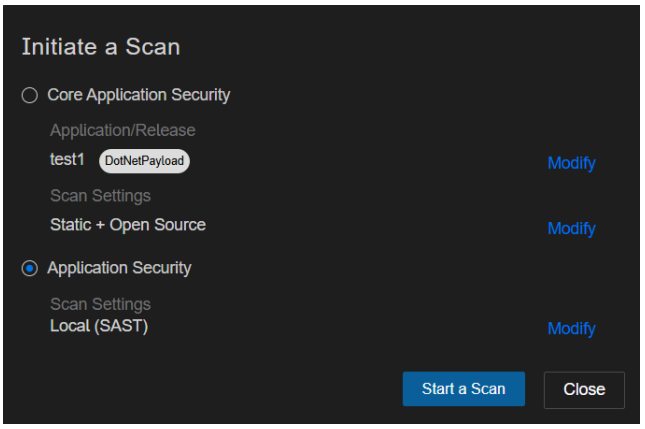
1. Click **Explorer** in the activity bar.
2. Navigate to the project on which you want perform the scan.
3. **Right-click** on a folder or file in the project.
4. Click **Start Scan with OpenText**.

   The Initiate a Scan box is displayed in the VS Code editor.



5. When you perform a new scan with OpenText™ Application Security Analysis Extension for Visual Studio Code, you must configure the scan settings for OpenText™ Core Application Security or Application Security.
    1. Select either **Core Application Security** or **Application Security** as needed.
    2. Click **Configure**. The OpenText™ Application Security Analysis Extension for Visual Studio Code navigates to the **Core Application Security** tab or **Application Security** tab in **OpenText™ Application Security Analysis** where you can configure the scan settings as needed.

       Once you have configure the scan settings for OpenText™ Core Application Security or Application Security, you can view the configuration in the Initiate a Scan box.



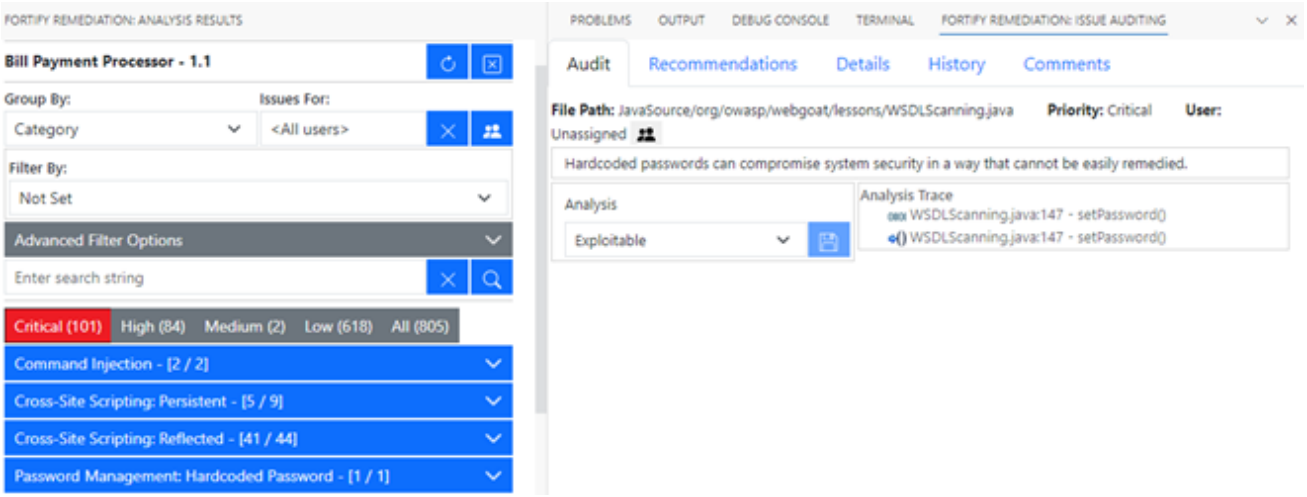
6. Select **Core Application Security** or **Application Security** as needed.
7. Click **Modify** if you want edit the scan settings for **Core Application Security** or **Application Security**.
8. Click **Start a Scan**.

OpenText Application Security Analysis packages the current project into a ZIP file, starts the scan, and displays the status information. When the scan is complete, OpenText Application Security Analysis displays the scan completion status in an information message.

1.4. Remediating your Code

<<This entire chapter is new for this release even though the text is not red. Please review completely.>>

After you open an application version on Application Security, the Fortify Extension for Visual Studio Code displays the analysis results in the **Analysis Results** view. This view displays all security issues, organized into tabs, which by default correspond to Fortify priority values. For example, the **Critical** folder contains all critical issues for a project and the **Low** folder contains all low-priority issues. Filters available for the application version determine which issues are visible. After you select an issue in the **Analysis Results** view, the **Issue Auditing** panel displays five tabs that provide information specific to the selected issue.



To remediate issues, the project you have open in VS Code must correspond to the application version you opened from Application Security (see Opening Application Security Application Versions).

This section contains the following topics:

- Opening Fortify Software Security Center Application Versions
- Viewing and Selecting Issues
- Grouping Issues
- Customizing Issue Visibility
- Searching for Issues
- Viewing Issue Information
- Locating Issues in your Source Code
- Adding Audit Information

1.4.1. Opening Fortify Software Security Center Application Versions

To view the analysis results, you must first connect to Application Security and open an application version.

To open an application version:

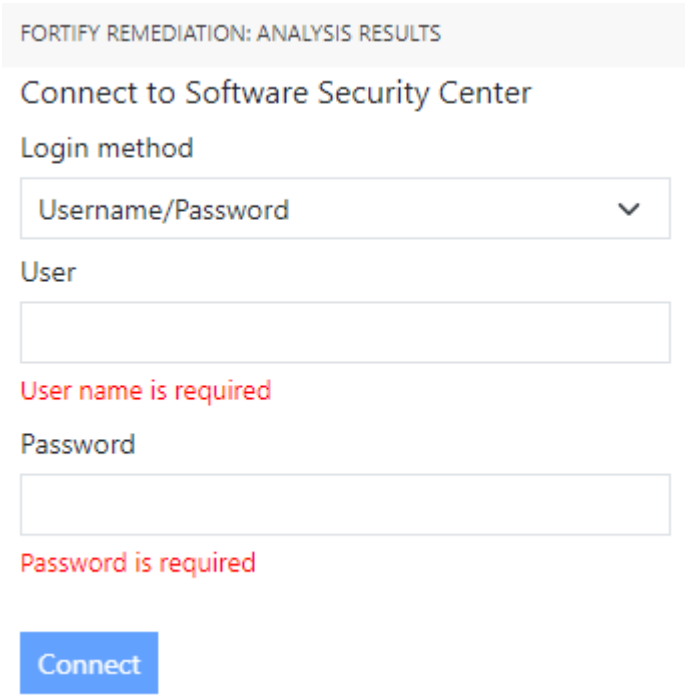
1. If the Fortify Remediation Extension for Visual Studio Code is not open, do one of the following:
   - Click **Fortify Remediation** in the activity bar. <<Temporary icon>>
   - If you have the Fortify Extension for Visual Studio installed, click **Fortify** in the activity bar, and then click **Remediation** in the side bar.

     If necessary, the extension is automatically installed.
2. Configure the Application Security connection settings in VS Code **Settings**:
   1. Open the VS Code **Settings** and search for `Fortify Remediation`.
   2. In the **Software Security Center URL** box, type the URL for your Application Security server.
   3. (Optional) From the **Login Method** list, select a default login method.
   4. (Optional) Select **Save Token** to save the authentication token value after a successful login for future connections to Application Security.

      Note

      For the **Username/Password** login method, the user name is always saved for future connections. The password is never saved.
3. Click **Fortify Remediation** in the activity bar.

   The **Analysis Results** view opens in the side bar.



4. From the **Login method** menu, select the login method set up for you in Application Security.
5. Depending on the selected login method, follow the procedure described in the following table.

Login method	Procedure
Username/Password	Type your Application Security user name and password.
Authentication Token	Specify the decoded value of a Application Security authentication token of type ToolsConnectToken.

6. Click **Connect** to connect to Application Security.

   The **Select Application Version** displays the application versions that your user account has permission to access.
7. Select an application name and version, and then click **Open**.

The Fortify Remediation Extension for Visual Studio Code displays the analysis results for the selected Application Security application version (see Viewing and Selecting Issues).

Note

To open a different application version on the same Application Security server to which you are already connected, click **Close application**  . To switch to a different Application Security instance, select **Log out**  and then reconnect to Application Security as described in this topic.

1.4.2. Viewing and Selecting Issues

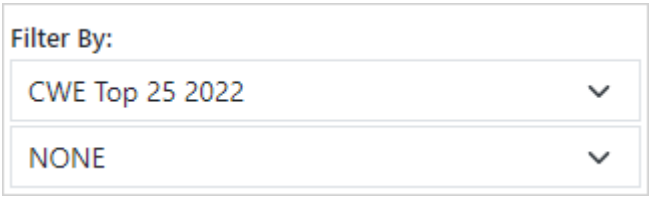
To view and select issues in an opened application version:

1. From the **Group By** list, select an attribute for sorting issues in all visible folders into groups.

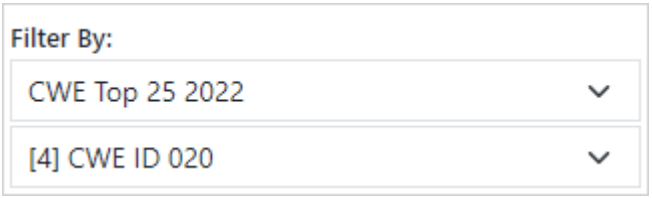
   The default grouping is **Category**. For a description of the available **Group By** attributes, see Grouping Issues.

2. By default, issues assigned to your Application Security user account are shown. From the **Issues For** list, you can do either of the following:
   - To show issues for all users, click **Clear** .
   - Select a Fortify Software Security Center user name.

3. To filter the issues within the selected grouping:
   1. From the **Filter By** list, select a filter category.

   2. To refine the issues further, select a filter option from the list below the selected filter category.

4. To apply a filter set to the issues, click **Advanced Filter Options**, and then from **Filter Set**, select one of the following filter sets to apply to issues:
   - Select **Security Auditor View** to list all issues relevant to a security auditor.
   - Select **Quick View** to list only issues in the **Critical** folder (these have a potentially high impact and a high likelihood of occurring) and the **High** folder (these have a potentially high impact and a low likelihood of occurring).

> **Note**
>
> The filter sets available depend on the issue template assigned to the application version you opened.

5. Click a tab to view the associated issues.

> **Note**
>
> The tabs shown depend on your **Group By**, **Issues For**, and **Filter Set** selections. It is possible that not all tabs are shown. The tabs shown also depend on the issue template associated with the application version.

   - The **Critical** tab contains issues that have a high impact and a high likelihood of exploitation. We recommend that you remediate critical issues immediately.
   - The **High** tab contains issues that have a high impact and a low likelihood of exploitation. We recommend that you remediate high issues with the next patch release.
   - The **Medium** tab contains issues that a have low impact and a high likelihood of exploitation. We recommend that you remediate medium issues as time permits.
   - The **Low** tab contains issues that have a low impact and a low likelihood of exploitation. We recommend that you remediate low issues as time permits (your organization can customize this category).
   - The **All** tab contains all issues.

   Within each tab, issues are grouped by the **Group By** selection. After each grouping name, enclosed in brackets, is the number of audited issues and the total number of issues in the group. For example, **Command Injection - [1 / 3]** indicates that one issue out of three categorized as Command Injection was audited.

6. Click to expand a grouping and view the issues it contains.

   The Fortify Extension for Visual Studio Code retrieves the corresponding issues from Application Security.

7. Select an issue to view its details in the **Issue Auditing** panel.

1.4.3. Grouping Issues

The issues visible in the **Analysis Results** view vary depending on the selected grouping attribute. The value you select from the **Group By** list sorts issues in all visible folders into subfolders. Use the **Group By** attributes to group and view the issues in different ways. The following table describes the available **Group By** attributes.

Attribute	Description
Analysis	Groups issues by the Analysis tag value assigned, such as Suspicious, Exploitable, and Not an Issue.
Analysis Type	Groups issues by analyzer product, such as SCA, WEBINSPECT, and SECURITYSCOPE (WebInspect Agent).
Analyzer	Groups issues by analyzer group, such as Control Flow, Data Flow, Pentest, and Structural.
App Defender Protected	Groups issues by whether Application Defender can protect the vulnerability category.
Category	Groups issues by vulnerability category. This is the default setting.
<custom_tagname>	Groups issues by the selected custom tag value assigned.
Correlated	Groups issues by whether the issue is related directly or indirectly with an issue uncovered by another analyzer. <<Removed from SSC>>
Correlation Group	Groups issues that are correlated with each other. <<Removed from SSC>>

Attribute	Description
Engine Priority	Groups issues based on the original priority value determined by the engine that identified the issue. **Note** This is only available in Application Security version 22.2.0 or later.
File Name	Groups issues by file name.
Folder	Groups issues by folders defined in the issue template.
Fortify Priority Order	Groups issues by Critical, High, Medium, and Low based on the issue priority.
Introduced date	Groups issues by the date the issue was first detected.
Issue State	Groups audited issues by whether the issue is an open issue or not an issue based on the level of analysis set for the primary tag. Values equivalent to Suspicious and Exploitable are considered open issue states.
Kingdom	Groups issues by the Seven Pernicious Kingdoms classification.

Attribute	Description
Manual	Groups issues by whether they were manually created by penetration test tools, and not automatically produced by a web crawler such as OpenText DAST.
<metadata_listname>	Groups issues using the alternative metadata external list names (for example, CWE, OWASP Top 10 *<year>*, PCI SSF *<version>*, STIG *<version>*, and others).
New Issue	Shows which issues are new since the last scan. For example, if you run a new scan, any issues that are new are displayed in the tree under the NEW group and the others are displayed in the UPDATED group. If removed issues are visible, issues not found in the latest scan are displayed in the REMOVED group.
Package	Groups issues by package or namespace. Nothing is shown for projects to which this option does not apply, such as C projects.
Primary Context	Groups issues where the primary location or sink node function call occurs in the same code context.
Priority Override	Groups issues by the Priority Override tag value assigned.

Attribute	Description
Sink	Groups issues that share the same dataflow sink function.
Source	Groups issues that share the same dataflow source functions.
Source Context	Groups dataflow issues that have the source function call contained in the same code context.
Source File	Groups dataflow issues by the source code file where the taint originated.
Status	Groups issues by the audit status (Reviewed, Unreviewed, or Under Review).
Taint Flag	Groups issues by the taint flags that they contain.
URL	Groups dynamic issues by the request URL.

1.4.4. Customizing Issue Visibility

You can customize the issues list in the **Analysis Results** view to determine which issues the Fortify Extension for Visual Studio Code displays.

To customize the display of hidden, removed, and suppressed issues:

1. In the **Analysis Results** view, expand the **Advanced Filter Options** section.
2. Under **Issue Visibility**, select or clear the following options:
    - To display all hidden issues, select **Show Hidden**.

      Note

      The visibility filter settings in the issue template associated with the application version determine which issues are hidden.

    - To display all the issues removed since the previous analysis, select **Show Removed**.
    - To display all suppressed issues, select **Show Suppressed**.

      Note

      Users who audit issues can suppress specific types of issues that are not considered high priority or of immediate concern. For example, auditors can suppress issues that are fixed, or issues that your organization plans not to fix.

The Fortify Remediation Extension for Visual Studio Code displays issues based on your selection.

Note

You can also specify issue visibility options in the Visual Studio Code Settings for **Fortify Remediation**.

1.4.5. Searching for Issues

You can use the search box above the issues list to search for issues. After you perform a search, the label next to the folder name changes to indicate the number of issues that match the search as a subset of the total.

To indicate the type of comparison to perform for a search in the **Analysis Results** view, wrap the search terms with delimiters. The following table shows the syntax to use for the search string.

Comparison	Description
contains	Searches for a term without any qualifying delimiters
equals	Searches for an exact match if the term is wrapped in quotation marks (`""`)
number range	Searches for a range of numbers using the standard mathematical interval notation of parentheses and/or brackets to indicate whether the endpoints are excluded or included, respectively. Example: (2,4] indicates greater than two and less than or equal to four
not equals	Excludes issues specified by the string by preceding the string with an exclamation character (!) Example, `file:!Main.java` returns all issues that are not in `Main.java`.

You can further qualify search terms with modifiers. The syntax for using a modifier is `modifier:<search_term>`. For more information, see Search Modifiers.

A search string can contain multiple modifiers and search terms. If you specify more than one modifier, the search returns only issues that match all the modified search terms. For example, `file:ApplicationContext.java category:SQL Injection` returns only SQL injection issues found in `ApplicationContext.java`.

If you use the same modifier more than once in a search string, then the search terms qualified by those modifiers are treated as an `OR` comparison. For example, `file:ApplicationContext.java category:SQL Injection category:Cross-Site Scripting` returns SQL injection issues and cross-site scripting issues found in `ApplicationContext.java`.

For complex searches, you can also insert the AND or the OR keyword between your search queries. Note that AND and OR operations have the same priority in searches.

This section contains the following topics:

1.4.5.1. Search Modifiers

You can use a search modifier to specify to which attribute of an issue the search term applies. To use a modifier that contains a space in the name, such as the name of the custom tag, you must enclose the modifier in brackets. For example, to search for issues that are new, type `[issue age]:new`.

A search that is not qualified by a modifier matches the search string based on the following issue attributes: kingdom, primary rule id, analyzer, filename, severity, class name, function name, instance id, package, confidence, type, subtype, taint flags, category, sink, and source.

The following examples describe using the search with and without applying a search modifier:

- To apply the search to all modifiers, type a string such as `control flow`. This searches all the modifiers and returns any result that contains the specified string.
- To apply the search to a specific modifier, type the modifier name and the string as follows: `analyzer:control flow`. This returns all results detected by the Control Flow Analyzer.

The following table describes the search modifiers. A few modifiers have a shortened modifier name indicated in parentheses. You can use either modifier string.

Search modifier	Description
accuracy	Searches for issues based on the accuracy value specified (0.1 through 5.0).
analysis	Searches for issues that have the specified audit analysis value, such as exploitable, not an issue, and so on.
[analysis type]	Searches for issues based on the analyzer product such as SCA and WEBINSPECT.
analyzer	Searches the issues for the specified analyzer such as control flow, data flow, structural, and so on.
[app defender protected] (def)	Searches for issues based on whether Application Defender can protect the vulnerability category (protected or not protected).
[attack payload]	Searches for issues that contain the search term in the part of the request that caused the vulnerability for penetration test results.
[attack type]	Searches for issues based on the type of penetration test attack conducted (URL, parameter, header, or cookie).

Search modifier	Description
audience	Searches for issues based on the intended audience, such as `dev`, `targeted`, `medium`, `broad`, and so on. **Note** **Caution** This metadata is legacy information that is no longer used and will be removed in a future release. Fortify recommends that you not use this search modifier.
audited	Searches for issues based on whether the primary tag is set (`true` or `false`). The default primary tag is the Analysis tag.
body	Searches for issues that contain the search term in the HTTP message body in penetration test results, which is all the data that is transmitted immediately following the headers.
category (cat)	Searches for the specified category or category substring.

Search modifier	Description
class	Searches for issues based on the specified class name.
comments (comment, com)	Searches for issues that contain the search term in the comments added to the issue.
commentuser	Searches for issues with comments from a specified user.
confidence (con)	Searches for issues that have the specified confidence value 0.1 through 5.0 (legacy metadata).
cookies	Searches for issues that contain the search term in the cookie from the HTTP query for penetration test results.
correlated	Searches for issues based on whether the issues are correlated with those detected by another analyzer.
[correlation group]	Searches for issues based on whether the issues are in the same correlation group.

Search modifier	Description
<custom_tagname>	Searches for issues based on the value of the specified custom tag. You can search a list-type custom tag using a range of values. The values of a list-type custom tag are an enumerated list where the first value is 0, the second is 1, and so on. You can use the search syntax for a range of numbers to search for ranges of list-type custom tag values. For example, `analysis:[0,2]` returns the issues that have the values of the first three analysis values, 0, 1, and 2 (Not an Issue, Reliability Issue, and Bad Practice). To search for a specific date in a date-type custom tag, specify the date in the format: yyyy-mm-dd. To search for issues that have no value set for a custom tag, use `<none>` as the search term. For example, to search for all issues that have no value set in the custom tag labeled Target Date, type: `[Target Date]:<none>`.
`dynamic`	Searches for issues that have the specified dynamic hot spot ranking value. <<not for SSC and thus not for remediation>>

Search modifier	Description
`[engine priority]`	Searches for issues based on the original priority value determined by the engine that identified the issue. **Note** This modifier is only available in Application Security version 22.2.0 or later.
`file`	Searches for issues where the primary location or sink node function call occurs in the specified file path.
`[fortify priority order]`	Searches for issues that have a priority level that matches the specified issue priority. Valid values are `critical`, `high`, `medium`, and `low`.
`headers`	Searches for issues that contain the search term in the request header for penetration test results.
`historyuser`	Searches for issues that have audit data modified by the specified user.
`[http version]`	Searches for issues based on the specified HTTP version such as `HTTP/1.1`.

Search modifier	Description
impact	Searches for issues based on the impact value specified (0.1 through 5.0).
[instance id]	Searches for an issue based on the specified instance ID.
[issue age]	Searches for the issue age, which is either new, updated, reintroduced, or removed.
[issue state]	Searches for audited issues based on whether the issue is an open issue or not an issue (determined by the level of analysis set for the primary tag).
kingdom	Searches for all issues in the specified kingdom.
likelihood	Searches for issues based on the specified likelihood value (0.1 through 5.0).
line	Searches for issues on the primary location line number. For dataflow issues, the value is the sink line number. Also see sourceline.
manual	Searches for issues that were manually created by penetration test tools, and not automatically produced by a web crawler such as Fortify WebInspect.

Search modifier	Description
`[mapped category]`	Searches for issues based on the specified category that is mapped across the various analyzers (OpenText SAST, OpenText DAST, and OpenText DAST Agent).
`maxconf`	Searches for all issues that have a confidence value equal to or less than the number specified as the search term.
`maxVirtConf`	Searches for dataflow issues that have a virtual call confidence value equal to or less than the number specified as the search term.
`<metadata_listname>`	Searches for issues based on the value of the specified metadata external list (for example, `[owasp top 10 <year>]`, `[cwe top 25 <year>]`, `[pci ssf <version>]`, `[stig <version>]`, and others).
`method`	Searches for issues based on the method, such as `GET`, `POST`, and so on.
`minconf`	Searches for all issues that have a confidence value equal to or greater than the number specified as the search term.

Search modifier	Description
min_virtual_call_confidence (virtconf, minVirtConf)	Searches for dataflow issues that have a virtual call confidence value equal to or greater than the number specified as the search term.
package	Searches for issues where the primary location occurs in the specified package or namespace. For dataflow issues, the primary location is the sink function.
parameters	Searches for issues that contain the search term in the HTTP query parameters.
primary	Searches for issues that have the specified primary tag value. By default, the primary tag is the Analysis tag.
[primary context]	Searches for issues where the primary location or sink node function call occurs in the specified code context. Also see sink and [source context].
primaryrule (rule)	Searches for all issues related to the specified sink rule.
probability	Searches for issues based on the probability value specified (1.0 through 5.0).

Search modifier	Description
[remediation effort]	Searches for issues based on the remediation effort value specified. The valid values are whole numbers from 1.0 to 12.0.
response	Searches for issues that contain the search term in the response from the protocol used in penetration test results.
severity (sev)	Searches for issues based on the specified severity value (legacy metadata).
sink	Searches for issues that have the specified sink function name. Also see [primary context].
source	Searches for dataflow issues that have the specified source function name. Also see [source context].
[source context]	Searches for dataflow issues that have the source function call in the specified code context. Also see source and [primary context].
sourcefile	Searches for dataflow issues with the source function call that the specified file contains. Also see file.

Search modifier	Description
`sourceline`	Searches for dataflow issues having taint source entering the flow on the specified line.
`status`	Searches issues that have the status `reviewed`, `not reviewed`, or `under review`.
`suppressed`	Searches for suppressed issues.
`taint`	Searches for issues that have the specified taint flag.
`trigger`	Searches for issues that contain the search term in the part of the response that shows that a vulnerability occurred for penetration test results.
`url`	Searches for issues based on the specified URL.
`user`	Searches for issues assigned to the specified user.

1.4.5.2. Search Query Examples

The following table contains search query examples.

Search task	Example query
Find privacy violations in file names that contain `jsp` with `getSSN()` as a source	`category:"privacy violation" source:getssn file:jsp`
Find file names that contain `com/test/123`	`file:com/test/123`
Find issues that contain `cleanse` as part of any modifier	`cleanse`
Find suppressed vulnerabilities with `asdf` in the comments	`suppressed:true comments:asdf`
Find all categories except for SQL Injection	`category:!SQL Injection`
Find issues that have a value specified for a custom tag labeled `version`	`version:!<none>`

1.4.6. Viewing Issue Information

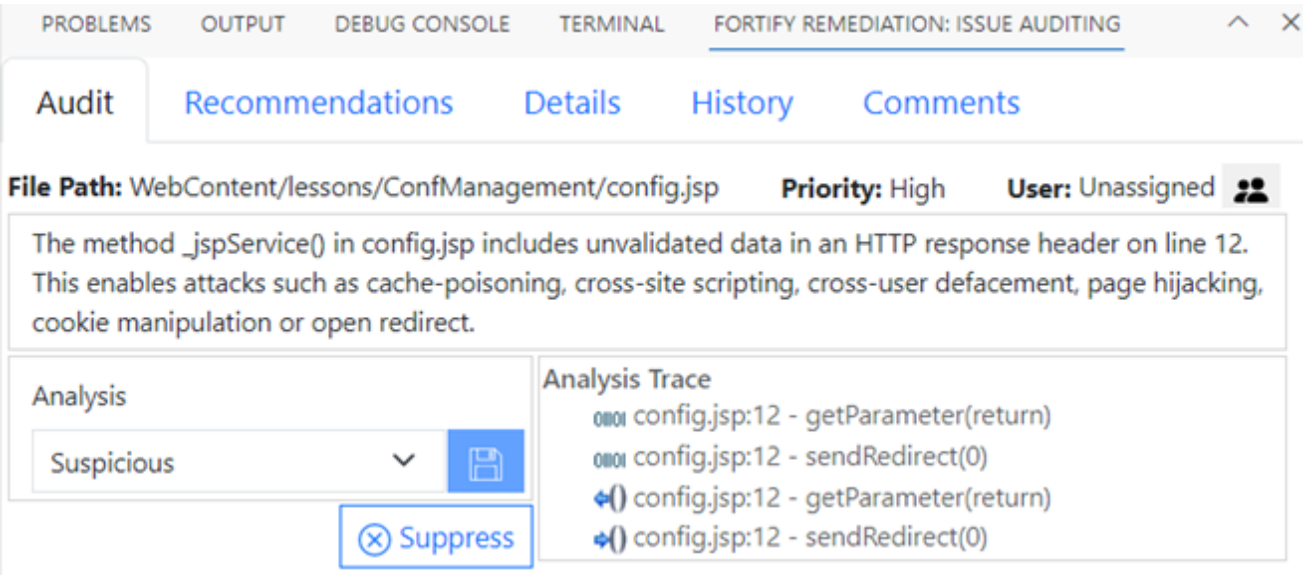
After you select an issue in the **Analysis Results** view, the Fortify Extension for Visual Studio Code displays the issue-specific content in the in the **Issue Auditing** panel on the **Audit**, **Recommendations**, **Details**, **History**, and **Comments** tabs.

This section contains the following topics:

1.4.6.1. Audit Tab

The **Audit** tab provides a dashboard of analysis information for the selected issue.

> **Note**
>
> Any changes you make on the **Audit** tab are automatically uploaded to the application version in Application Security.



The following table describes the **Audit** tab features.

Element	Description
Priority	The Fortify priority value determined for the selected issue. If Application Security has Priority Override enabled and the priority value was changed, then the current priority value is displayed with the original Fortify priority value in parentheses. For instructions on how to change the priority override, see Assigning a Tag an Issue.
User	User assigned to the selected issue. For instructions on how to assign a user to an issue, see Assigning Users to Issues.
Analysis	Your assessment of the selected issue. To change the assessment, select an item from the list. This is the primary tag defined in Application Security for the application version. The default primary tag is **Analysis**, but your organization might have a different tag designated as the primary tag.

Element	Description
<custom_tagname>	Any custom tags your organization has defined in Application Security. If available, these are displayed below the primary tag. For information on how to make changes to these tags, see Assigning Tags to Issues. If the audit results have been submitted to Fortify Audit Assistant in Application Security, then in addition to any other custom tags, the tab displays the following tags: • **AA_Prediction**—Exploitability level that Fortify Audit Assistant assigned to the issue. You cannot modify this tag value. • **AA_Confidence**—Confidence level from Fortify Audit Assistant for the accuracy of its **AA_Prediction** value. This is a percentage expressed in values that range from 0.000 to 1.000. For example, a value of 0.982 indicates a confidence level of 98.2 percent. You cannot change this tag value. • **AA_Training**—Whether to include or exclude the issue from Fortify Audit Assistant training. You can modify this value For more information about Fortify Audit Assistant, see the *OpenText™ Application Security User Guide* in Fortify Software Security Center Documentation.

Element	Description
Suppress	Suppresses the issue. For information about suppressing issues, see Suppressing Issues.
Analysis Trace	Items of evidence that the analyzer uncovered. The analysis trace evidence is presented in the order it was discovered. For descriptions of the analysis trace icons, see Analysis Trace.

See Also

Updating Audit Information

1.4.6.2. Analysis Trace

The analysis trace on the **Audit** tab is presented in sequential order. For dataflow issues, this trace is a presentation of the path that the tainted data follows from the source function to the sink function. For example, when you select an issue that is related to potentially tainted dataflow, the analysis trace box shows the direction of the dataflow in this section of the source code.

The analysis trace box uses the icons described in the following table to show how the dataflow moves in this section of the source code or execution order.

Icon	Description
	Data is assigned to a field or variable
	Information is read from a source external to the code (HTML form, URL, and so on)
	Data is assigned to a globally scoped field or variable
	A comparison is made
	The function call receives tainted data
	The function call returns tainted data

Icon	Description
	Passthrough, tainted data passes from one parameter to another **Note** This is typically shown as functionA(x : y) to indicate that data is transferred from x to y. The x and y values are one of the following: • An argument index • `return` —The return value of a function • `this` —The instance of the current object • A specific object field or key
	An alias is created for a memory location
	Data is read from a variable
	Data is read from a global variable
	Tainted data is returned from a function
	A pointer is created

Icon	Description
	A pointer is dereferenced
	The scope of a variable ends
	The execution jumps
	A branch is taken in the code execution
	A branch is not taken in the code execution
	Generic
	A runtime source, sink, or validation step
	Taint change

The analysis trace box can contain inductions. Inductions provide supporting evidence for their parent nodes. Inductions consist of:

- A text node displayed in italics as a child of the trace node. This text node is expanded by default.
- An induction trace, displayed as a child of the text node (a box surrounds the induction trace).

The italics and the box distinguish the induction from a standard subtrace. To display the induction reference information for that induction, click it.

1.4.6.3. Recommendations Tab

The **Recommendations** tab provides suggestions and examples on how to secure a vulnerability or remedy a bad practice. The following table describes the sections on this tab.

Section	Description
Recommendations/Custom Recommendations	Describes possible solutions for the selected issue. It can also include examples and recommendations defined by your organization.
Tips/Custom Tips	Provides useful information specific to the selected issue, and any custom tips defined by your organization.
References/Custom References	Lists references for the recommendations provided, including any custom references defined by your organization.

1.4.6.4. Details Tab

The **Details** tab provides an abstract of the selected issue description, a detailed explanation, and examples. The following table describes the sections on this tab.

Section	Description
Abstract/Custom Abstract	Summary of the selected issue, including any custom abstracts defined by your organization.
Explanation/Custom Explanation	Description of the conditions under which an issue of the selected type occurs. This includes a discussion of the vulnerability, the constructs typically associated with it, ways in which attackers can exploit it, and the potential ramifications of an attack. This section also includes any custom explanations defined by your organization.
Instance ID	Unique identifier for the issue.
Primary Rule ID	Identifier for the primary rule used to uncover the issue.
Priority Metadata Values	Priority metadata values for this issue including impact and likelihood.
Legacy Priority Metadata Values	Legacy priority metadata values for the issue including severity and confidence.

1.4.6.5. History Tab

The **History** tab displays a history of audit actions, including details such as the time and date, and the name of the user who modified the issue.

1.4.6.6. Comments Tab

The **Comments** tab displays all comments that were submitted for the issue. To add a comment for an issue, see Adding Comments to Issues.

1.4.7. Locating Issues in your Source Code

You can use the Fortify Remediation Extension for Visual Studio Code to locate security-related issues in your code. Make sure that the revision of the source code open in VS Code corresponds to the application version you opened on Application Security.

To locate issues in the source code, do one of the following:

- Select an issue in the **Analysis Results** view.
- From the **Audit** tab, select a line in the Analysis Trace.

VS Code places the focus on the line of code that contains the selected security-related issue.

1.4.8. Adding Audit Information

After you select and review an issue, you can add audit information on the **Audit** tab. To see any updates to the audit results made in Application Security, click **Refresh**  .

This section contains the following topics:

1.4.8.1. Assigning Users to Issues

To assign a user to an issue:

1. From the **Analysis Results** view in the side bar, select an issue.
2. In the **Issue Auditing** panel, click the **Audit** tab.
3. Click **Select User**, select a user name, and then click **Save**.

   To leave the issue unassigned, click **Unassign User**.

The Fortify Remediation Extension for Visual Studio Code makes the update to the application version in Application Security.

1.4.8.2. Assigning a Tag to an Issue

To assign a custom tag value to an issue:

1. From the **Analysis Results** view, select an issue.
2. From the **Analysis** list on the **Audit** tab, select a value that reflects your evaluation of this issue.

   This is the primary tag as defined in Application Security. The default primary tag is **Analysis**, but your organization might have a different tag designated as the primary tag.
3. If the priority override capability is enabled on Application Security, you can override the priority value for the issue as follows:
   1. From the **Priority Override** list, select the preferred priority value.
   2. Explain why you changed the value in the comment box outlined in red.
4. If custom tags defined for the project exist, provide values for them.

   The Fortify Remediation Extension for Visual Studio Code displays all custom tags assigned to the application; however, you can only provide values for tags that your Application Security user account has permission to edit.

   Use the following instructions to provide a value for a custom tag:
   - For text- and decimal-type custom tags, type the value in the box. Text-type custom tags accept up to 500 characters (HTML/XML tags and newlines are not allowed).
   - For date-type custom tags, type a date or click **Select Date** 🗔 to select a date from a calendar.
5. If a tag requires a comment, then after you provide a value for the tag, then you must type a comment in the box outlined in red.
6. Click Save 💾.

The Fortify Remediation Extension for Visual Studio Code makes the updates to the application version in Application Security.

1.4.8.3. Adding Comments to Issues

The comments tab in the **Fortify Remediation: Issue Auditing** panel displays any comments submitted for the selected issue.

To add a comment to an issue:

1. From the **Analysis Results** view in the side bar, select an issue.
2. In the **Issue Auditing** panel, click the **Comments** tab.
3. In the **Enter comment** box, type your comment.
4. Click **Save**.

The Fortify Extension for Visual Studio Code makes the update to the application version in Application Security.

1.4.8.4. Suppressing Issues

You can suppress issues that are either fixed or that you do not plan to fix. Suppression marks the issue and all future discoveries of this issue as suppressed. As such, it is a semi-permanent marking of a vulnerability.

To suppress an issue:

1. In the **Analysis Results** view, select the issue.
2. In the **Issue Auditing** panel, click the **Audit** tab, and then click **Suppress**.

By default, Fortify Remediation Extension for Visual Studio Code automatically refreshes the **Analysis Results** view issue list and hides suppressed issues.

To display issues that have been suppressed:

- In the **Analysis Results** view, expand **Advanced Filter Options**, and then select **Show Suppressed**.

  Note

  You can review suppressed issues by searching for them using the `suppressed` search modifier.

To unsuppress an issue, first display the suppressed issues, and then do the following:

1. In the **Analysis Results** view, select the suppressed issue.

   Each suppressed issue is tagged with an "S" icon.

2. In the **Issue Auditing** panel, click the **Audit** tab, and then click **Unsuppress**.

© Copyright 2025 Open Text

For more info, visit https://docs.microfocus.com