

OpenText™ Core Application Security Remediation Extension for Visual Studio Code

Software Version: 25.2

User Guide

Document Release Date: May 2025

Software Release Date: May 2025

Legal Notices

Open Text Corporation

275 Frank Tompa Drive, Waterloo, Ontario, Canada, N2L 0A1

Copyright Notice

Copyright 2025 Open Text.

The only warranties for products and services of Open Text and its affiliates and licensors (“Open Text”) are as may be set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. Open Text shall not be liable for technical or editorial errors or omissions contained herein. The information contained herein is subject to change without notice.

Trademark Notices

“OpenText” and other Open Text trademarks and service marks are the property of Open Text or its affiliates. All other trademarks or service marks are the property of their respective owners.

Documentation updates

The title page of this document contains the following identifying information:

- Software Version number
- Document Release Date, which changes each time the document is updated
- Software Release Date, which indicates the release date of this version of the software

This document was produced for OpenText™ Core Application Security Remediation Extension for Visual Studio Code CE 25.2 on May 23, 2025.

Contents

Preface	4
Contacting Customer Support	4
For more information	4
Product feature videos	4
Getting started	5
Product name changes	5
Requirements for using the Core Application Security Remediation Extension for Visual Studio Code	5
Installing the Core Application Security Remediation Extension for Visual Studio Code	6
Configuring the Core Application Security Remediation Extension for Visual Studio Code	6
Remediating your code	7
Opening application releases	7
View and selecting issues	8
Grouping issues	10
Auditing issues	11
Analysis trace	13
Analysis Trace Icons	13
Vulnerability	14
Recommendations	14
History	15
Auditing multiple issues	15
Send Documentation Feedback	17

Preface

Contacting Customer Support

Visit the [Customer Support](#) website to:

- Manage licenses and entitlements
- Create and manage technical assistance requests
- Browse documentation and knowledge articles
- Download software
- Explore the Community

For more information

For more information about OpenText Application Security Testing products, visit [OpenText Application Security](#).

Product feature videos

You can find videos that highlight OpenText Application Security Software products and features on the [Fortify Unplugged YouTube™ channel](#).

Getting started

This guide describes how to install the OpenText™ Core Application Security Remediation Extension for Visual Studio Code and use it to upload code for static analysis and open analysis results for remediation.

This section contains the following topics:

Product name changes	5
Requirements for using the Core Application Security Remediation Extension for Visual Studio Code	5
Installing the Core Application Security Remediation Extension for Visual Studio Code	6
Configuring the Core Application Security Remediation Extension for Visual Studio Code	6

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 WebInspect	OpenText™ Dynamic Application Security Testing (OpenText DAST)
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.

Requirements for using the Core Application Security Remediation Extension for Visual Studio Code

To use the Core Application Security Remediation Extension for Visual Studio Code, you must have the following:

- An API root URL.
For a list of data center API root URLs, see the *OpenText Core Application Security User Guide*.
- Your OpenText Core Application Security login credentials, personal access token, or authentication token for your OpenText Core Application Security account.

Installing the Core Application Security Remediation Extension for Visual Studio Code

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

After you install the Core Application Security Remediation Extension for Visual Studio Code, the Visual Studio menu bar now includes the OpenText Core Application Security Remediation menu.

Configuring the Core Application Security Remediation Extension for Visual Studio Code

To configure the Core Application Security Remediation Extension for Visual Studio Code connection settings:

1. Open VS Code **Settings** and search for OpenText Core Application Security Remediation.
2. Configure the following properties for OpenText Core Application Security Remediation:

Property	Description
Opentext core: Login Method	Select a default login method to connect to your Core Application Security account. You can select one of the following options: <ul style="list-style-type: none">• Username/Password• Personal Access Token• Authentication Token
Opentext core: API URI	Specify the API root URL for the Core Application Security portal. For more information about API root URLs, see <i>Core</i>

Property	Description
	<i>Application Security</i> documentation.
Opentext core: Enable Debug level log	If you encounter any errors, you can enable debug mode to help troubleshoot. When you enable debug mode, Core Application Security Remediation Extension for Visual Studio Code writes additional information to the log files. Click the here button to view the log file with the Month-Year file name format.

Remediating your code

After you select an application release on Core Application Security, the Core Application Security Remediation Extension for Visual Studio Code displays the issues in the **Results** view. This view displays all security issues, organized into tabs, which by default correspond to Core Application Security priority values.

To remediate issues, the project you have open in VS Code must correspond to the application release you opened from Core Application Security (see "[Opening application releases](#)" below).

This section contains the following topics:

Opening application releases	7
View and selecting issues	8
Auditing issues	11

Opening application releases

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

To open an application release:

1. Open VS Code and click **OpenText Core Application Security Remediation** in the activity bar. The **OpenText Core Application Security Remediation** login page appears.
2. From the **Login Method** list, select the login method set up for you in Core Application Security.

3. Specify the following values based on the selected login method.

Login method	Procedure
Username/Password	Enter the Core Application Security user name, password, and tenant ID.
Personal Access Token	<p>Enter the Core Application Security username, access token, and tenant ID.</p> <p>Note: Access token refers to the personal access token generated in Core Application Security. For instructions on how to create a personal access token in Core Application Security, see the <i>OpenText™ Core Application Security User Guide</i></p>
Authentication Token	<p>Specify the API key and API secret in the Client Id and Client Secret text boxes respectively to connect to Core Application Security.</p> <p>Note: For instructions on how to create an API key and secret in Core Application Security, see the <i>OpenText™ Core Application Security User Guide</i></p>

4. Click **Sign In** to connect to Core Application Security.
The **Select Application & Release** list includes the application and releases that your user account has permission to access.
5. Select an application and release from the **Application** and **Release** drop-down lists respectively, and then click **Show Results**.

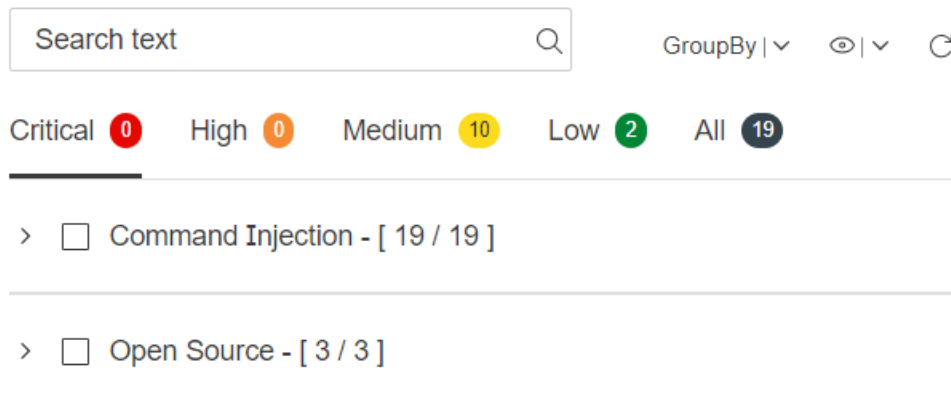
The Core Application Security Remediation Extension for Visual Studio Code displays the Results for the selected Core Application Security application release (see "[View and selecting issues](#)" below).

View and selecting issues

To view and select issues in an opened application release:

1. From the **GroupBy** 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](#)" on page 10.

- By default, only open issues assigned to your Core Application Security user account are shown. Click a tab to view the associated issues.

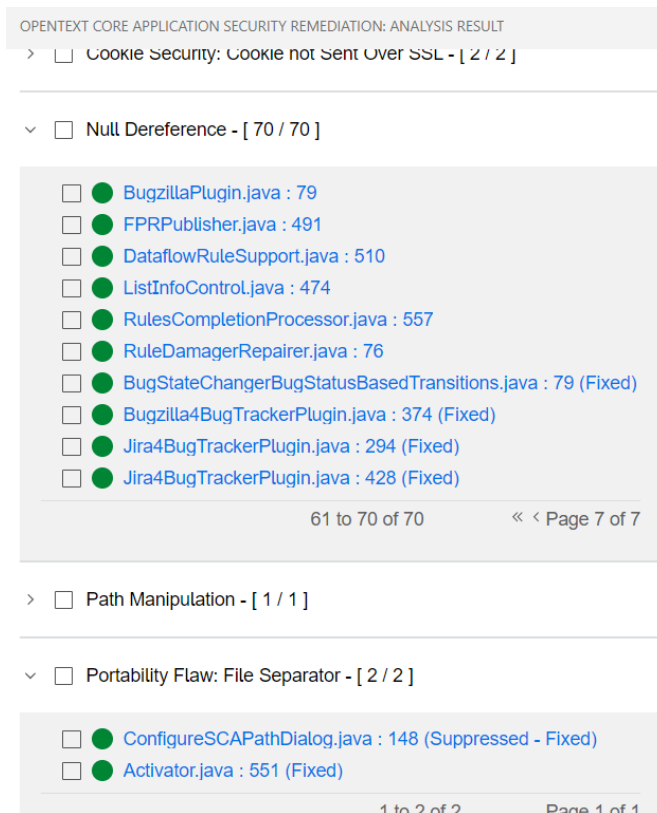


Note: The tabs shown depend on your **Group By** selection. It is possible that not all tabs are shown. The tabs shown also depend on the issue template associated with the application release.

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

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

To view fixed and/or suppressed issues, click **Fixed** and/or **Suppressed** from the **Visibility** (👁) list. The fixed and/or suppressed issues are indicated with a Fixed, Suppressed, or Suppressed-Fixed, enclosed in brackets, in the **Results** view.



You can also search for an issue category or issues within a tab in the **Search text** box.

4. Select an issue to view its details in the **OpenText Core Application Security Remediation** tab.
5. Click the **Refresh** icon to sync any changes to the issues in the application release in Core Application Security.

Grouping issues

The issues visible in the **Results** view vary depending on the selected grouping attribute. The value you select from the **GroupBy** 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	Values
Category	Groups issues by vulnerability category. This is the default setting.	
Auditor Status	Groups issues by Auditor issue remediation status.	<ul style="list-style-type: none">Unsuppressed system values: Pending Review, Remediation

Attribute	Description	Values
		Required, Remediation Deferred, Risk Mitigated, Suspicious, Proposed Not an Issue Note: Suspicious and Proposed Not an Issue are set by SAST Aviator. <ul style="list-style-type: none">• Suppressed system values: Risk Accepted, Not an Issue• User-defined
Severity	Groups issues by issue severity.	Critical, High, Medium, Low, Best Practice, Info
Fortify Aviator	Groups issues audited by SAST Aviator.	False, True
Status	Groups issues by issue status.	New, Existing, Reopen, Fixed/Fix Validated

Auditing issues


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

If you have the Edit Issues permission, you can assign a user, set the developer status, and add comments for issues in the Audit Summary view. If you have the Audit Issues permission, you can also edit the issue's auditor status and severity.

Note: Any changes you make on the **Audit** tab are automatically uploaded to the application release in Core Application Security.

Audit

Status	Introduced Date	Last Found Date
New	2023/08/07	2023/08/07

 Save

Auditor Status

Select your option

Developer Status

Select your option

Assigned User

Select your option

Severity

Comment

Add comment here

To audit an issue:

1. Make sure that the **Audit Summary** view is open.
2. From the issues list in the **Results**, select an issue. You can select multiple issues in the **Results** view to make the same edits to multiple issues.
3. In the **Audit** view, select the user to assign to the issue from **Assigned User** list.
4. To change the issue's development status, select the status from the **Developer Status** list
5. To change the auditor status, select the status from the **Auditor Status** list.
6. To change the issue severity, select an issue severity from the **Severity** list.
7. To add a comment for the issue, type your comment in the box at the bottom of the **Comments** area..
8. Click **Save**.

The changes and comments are displayed in the **History** tab.

The Core Application Security Remediation Extension for Visual Studio Code saves your changes for the selected application release.

See Also

["Analysis trace" on the next page](#)

["Vulnerability" on page 14](#)

["Recommendations" on page 14](#)

["History" on page 15](#)















Analysis trace







The **Analysis Trace** tab on the **OpenTextCore Application SecurityRemediation** tab displays the relevant trace output. This is a set of program points that show how the analyzer found the issue. VS Code places the focus on the line of code that contains the selected security-related issue.

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.

Analysis Trace Icons

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

Icon	Description	Icon	Description
	Data is assigned to a field or variable		Tainted data is returned from a function
	Information is read from a source external to the code such as an HTML form or a URL		A pointer is created
	Data is assigned to a globally scoped field or variable		A pointer is dereferenced
	A comparison is made		The scope of a variable ends
	The function call receives tainted data		The execution jumps
	The function call returns tainted data		A branch is taken in the code execution
	Passthrough, tainted data passes from one place 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:		A branch is not taken in the code execution

Icon	Description	Icon	Description
	<ul style="list-style-type: none">• An argument index• <code>return</code>—The return value of a function• <code>this</code>—The instance of the current object• A specific object field or key		
	An alias is created for a memory location		Generic
	Data is read from a variable		A runtime source, sink, or validation step
	Data is read from a global variable		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.

Vulnerability

The **Vulnerability** tab displays the following technical details about the issue: issue summary; explanation of the execution and implications of the issue; instance ID and primary rule ID; and standards and best practices information from Fortify Software Security Research.

Recommendations

The **Recommendations** tab displays remediation information and references for further research. 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.

Section	Description
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.

History

The **History** tab displays a log of the following issue events: audit changes, comments, and system events (status changes, copy state actions). You can filter the log by the event type (audit, comment, or system event). You can also refresh the history to fetch the latest events.

Auditing multiple issues

You can select multiple issues in an application release and bulk audit multiple issues.

To bulk audit multiple issues:

1. Open an application release in the Results view (see ["Opening application releases" on page 7](#)).
2. In the Results view, select the check boxes next to the issues.

The **OpenText Core Application Security Remediation** tab displays the selected issues under the **Selected Issues** area.

The **Selected Issues** area displays the total number of selected issues, issue ID, release, primary location, and scan tool.

The screenshot displays the OpenText Core Application Security Remediation interface. On the left, a sidebar shows a list of issues categorized by severity (Critical, High, Medium, Low, All) and type (Cross-Site Scripting, HTML5, JSON Injection). The main area shows a code editor with a Java file. On the right, the 'Selected Issues' panel is active, displaying a table of selected issues. Below the table, there is an 'Audit' panel with fields for Auditor Status, Developer Status, Assigned User, Severity, and a Comment field.

Issue ID	Release	Primary Location	S	T
265742271	WebGoatJavaPayload	UserController.java : 547	S	
265742295	WebGoatJavaPayload	ProductController.java : 93	S	
265742277	WebGoatJavaPayload	WebSecurityConfiguration.java : 104	S	
265742256	WebGoatJavaPayload	WebSecurityConfiguration.java : 148	S	

3. Click an issue ID under the **Issue ID** column to view the audit information, analysis trace, vulnerability, recommendations, and history for each individual issue.

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS OPENTEXT CORE APPLICATION SECURITY REMEDIATION

WebGoatJavaPayload Critical Category : Cross-Site Scripting: Reflected

< Multi Selection 265742271 src/main/java/com/microfocus/example/web/controllers/UserController.java : 547

Analysis Trace Vulnerability Recommendations History

➔() UserController.java:543 - serveXMLFile(0)

➔() UserController.java:545 - loadAsResource(0 : return)

➔() FileSystemStorageService.java:148 - loadAsResource(0 : return)

➔() FileSystemStorageService.java:159 - get(0 : return)

:= FileSystemStorageService.java:159 - Assignment to file

➔() FileSystemStorageService.java:164 - toUri(this : return)

➔() FileSystemStorageService.java:164 - UrlResource(0 : this)

:= FileSystemStorageService.java:164 - Assignment to resource

4. Click the **Multi Selection** button to navigate to the previous page.
5. In the audit panel, edit the fields as needed. The following fields are available for editing:
Assigned User, Developer Status, Auditor, Status, Severity, and Comments.
6. Click **Save**.
7. Click an issue ID under the **Issue ID** column to view the audit information, analysis trace, vulnerability, recommendations, and history for individual issues.

Send Documentation Feedback

If you have comments about this document, you can [contact the documentation team](#) by email.

Note: If you are experiencing a technical issue with our product, do not email the documentation team. Instead, contact Customer Support at <https://www.microfocus.com/support> so they can assist you.

If an email client is configured on this computer, click the link above to contact the documentation team and an email window opens with the following information in the subject line:

Feedback on User Guide (Core Application Security Remediation Extension for Visual Studio Code 25.2)

Just add your feedback to the email and click send.

If no email client is available, copy the information above to a new message in a web mail client, and send your feedback to fortifydocteam@opentext.com.

We appreciate your feedback!