IBM Security AppScan Source for Analysis
Version 9.0.0.1

User Guide

IBM

IBM Security AppScan Source for Analysis
Version 9.0.0.1

User Guide

Contents

Chapter 1. Introduction to AppScan Source for Analysis

This section describes how AppScan® Source for Analysis fits into the total AppScan Source solution and provides a basis for understanding the software assurance workflow.

Introduction to IBM Security AppScan Source

IBM® Security AppScan Source delivers maximum value to every user in your organization who plays a role in software security. Whether a security analyst, quality assurance professional, developer, or executive, the AppScan Source products deliver the functionality, flexibility, and power you need - right to your desktop.

The product set includes:

- **AppScan Source for Analysis**: Workbench to configure applications and projects, scan code, analyze, triage, and take action on priority vulnerabilities.
- **AppScan Source for Automation**: Allows you to automate key aspects of the AppScan Source workflow and integrate security with build environments during the software development life cycle.
- **AppScan Source for Development**: Developer plug-ins integrate many AppScan Source for Analysis features into Microsoft Visual Studio, the Eclipse workbench, and Rational® Application Developer for WebSphere® Software (RAD). This allows software developers to find and take action on vulnerabilities during the development process. The Eclipse plug-in allows you to scan source code for security vulnerabilities and, optionally, scan for and quality risks as well as create quality rule configuration files that enable quality scanning in the AppScan Source command line interface (CLI) and AppScan Source for Automation. In addition, IBM Worklight® projects can be scanned with the Eclipse plug-in.

To enhance the value of AppScan Source within your organization, the products include these components:

- **AppScan Source Security Knowledgebase**: In-context intelligence on each vulnerability, offering precise descriptions about the root cause, severity of risk, and actionable remediation advice.
- **AppScan Enterprise Server**: Most AppScan Source products and components must communicate with an AppScan Enterprise Server. Without one, you can use AppScan Source for Development in local mode - but features such as custom rules, shared scan configurations, and shared filters will be unavailable.

  The server provides centralized user management capabilities and a mechanism for sharing assessments via the AppScan Source Database. The server includes an optional Enterprise Console component. If your administrator installs this component, you can publish assessments to it from AppScan Source for Analysis, AppScan Source for Automation, and the AppScan Source command line interface (CLI). The Enterprise Console offers a variety of tools for working with your assessments - such as reporting features, issue management, trend analysis, and dashboards.

  Note:

  – AppScan Enterprise Server is not supported on OS X.

  - If you have a basic server license, the server may only be accessed by up to ten (10) concurrent connections from AppScan products. With a premium server license, unlimited connections are allowed.

Important: When scanning, AppScan Enterprise Server and AppScan Source clients (except AppScan Source for Development) both require a direct connection to the AppScan Source Database (either solidDB® or Oracle).

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

Translated national languages

The AppScan Source user interfaces are available in these languages:
- English
- Brazilian Portuguese
- Simplified Chinese
- Traditional Chinese
- German
- Spanish
- French
- Italian
- Japanese
- Korean

United States government regulation compliance

Compliance with United States government security and information technology regulations help to remove sales impediments and roadblocks. It also provides a proof point to prospects worldwide that IBM is working to make their products the most secure in the industry. This topic lists the standards and guidelines that AppScan Source supports.
- "Internet Protocol Version 6 (IPv6)"
- "Federal Information Processing Standard (FIPS)"
- "National Institute of Standards and Technology (NIST) Special Publication (SP) 800-131a" on page 3
- "Windows 7 machines that are configured to use the United States Government Configuration Baseline (USGCB)" on page 4

Internet Protocol Version 6 (IPv6)

AppScan Source is enabled for IPv6, with these exceptions:
- Inputting IPv6 numerical addresses is not supported and a host name must be entered instead. Inputting IPv4 numerical addresses is supported.
- IPv6 is not supported when connecting to Rational Team Concert™.

Federal Information Processing Standard (FIPS)

On Windows and Linux platforms that are supported by AppScan Source, AppScan Source supports FIPS Publication 140-2, by using a FIPS 140-2 validated cryptographic module and approved algorithms. On OS X platforms that are supported by AppScan Source, manual steps are needed to operate in FIPS 140-2 mode.

To learn background information about AppScan Source FIPS compliance - and to learn how to enable and disable AppScan Source FIPS 140-2 mode, see these technotes:

- Operating AppScan Source version 8.7 or later in FIPS 140-2 mode on OS X
- How to enable/disable/verify FIPS 140-2 mode in AppScan Source (Linux and Windows)
- Background information about AppScan Source version 8.7 or later FIPS 140-2 support

National Institute of Standards and Technology (NIST) Special Publication (SP) 800-131a

NIST SP 800-131A guidelines provide cryptographic key management guidance. These guidelines include:

- Key management procedures.
- How to use cryptographic algorithms.
- Algorithms to use and their minimum strengths.
- Key lengths for secure communications.

Government agencies and financial institutions use the NIST SP 800-131A guidelines to ensure that the products conform to specified security requirements.

NIST SP 800-131A is supported only when AppScan Source is operating in FIPS 140-2 mode. To learn about enabling and disabling AppScan Source FIPS 140-2 mode, see "Federal Information Processing Standard (FIPS)" on page 2.

Important: If the AppScan Enterprise Server that you will connect to is enabled for NIST 800-131a compliance, you must set AppScan Source to force Transport Layer Security V1.2. If Transport Layer Security V1.2 is not forced, connections to the server will fail.

- If you are **not** installing the AppScan Source Database (for example, you are only installing client components), you can force Transport Layer Security V1.2 by modifying `<data_dir>\config\ounce.ozsettings` (where `<data_dir>` is the location of your AppScan Source program data, as described in "Installation and user data file locations" on page 258)). In this file, locate this setting:

  ```
  <Setting
     name="tls_protocol_version"
     read_only="false"
     default_value="0"
     value="0"
     description="Minor Version of the TLS Connection Protocol"
     type="text"
     display_name="TLS Protocol Version"
     display_name_id=""
     available_values="0:1:2"
     hidden="false"
     force_upgrade="false"
   />
  ```

  In the setting, change `value="0"` to `value="2"` and then save the file.

- If you are installing the AppScan Source Database, you force Transport Layer Security V1.2 in the IBM Security AppScan Enterprise Server Database Configuration tool after installing both AppScan Source and the Enterprise Server.

Windows 7 machines that are configured to use the United States Government Configuration Baseline (USGCB)

AppScan Source supports scanning applications on Windows 7 machines that are configured with the USGCB specification.

Note: On machines that are configured with the USGCB specification, AppScan Source does not support defect tracking system integration with HP Quality Center or Rational ClearQuest®.

What's New in AppScan Source

This topic describes new features that have been added to AppScan Source.

What's New in AppScan Source Version 9.0.0.1

- “New platform and integration solution support”
- “Improved JavaScript Statement Graph”

New platform and integration solution support

As of AppScan Source Version 9.0.0.1:

- Visual Studio 2013 project files can be scanned on Windows - and the AppScan Source for Development (Visual Studio plug-in) can be applied to Visual Studio 2013 on Windows.

Improved JavaScript Statement Graph

JavaScript statements in the Trace view now include the section of code that is of interest, if available.

What's New in AppScan Source Version 9.0

- “New platform and integration solution support”
- “IBM Worklight integration” on page 5
- “Using AppScan Source for Development without an AppScan Enterprise Server” on page 5
- “Optional AppScan Source for Development Eclipse plug-in quality component” on page 6
- “Floating license option for AppScan Source for Automation” on page 6
- “Enhanced and new scanning support” on page 6
- “Windows 7 machines that are configured to use the United States Government Configuration Baseline (USGCB)” on page 6
- “Quality analysis feature deprecation in Version 9.0” on page 6

New platform and integration solution support

As of AppScan Source Version 9.0, these operating systems are supported:

- Microsoft Windows 8 Professional and Enterprise

- Microsoft Windows 8.1 Professional and Enterprise
- Microsoft Windows Server 2012 R2 Datacenter, Standard, and Essentials Editions
- Red Hat Enterprise Linux Version 6 Update 5

In addition:

- **OS X**: The AppScan Source for Development Eclipse plug-in is now supported on OS X:
  - Eclipse Versions 3.6, 3.7, 3.8, 4.2, 4.2.x, 4.3, 4.3.1, and 4.3.2 project files and workspaces (Java™ and IBM Worklight only) can be scanned - and the AppScan Source for Development (Eclipse plug-in) can be applied to these versions of Eclipse.
  - Rational Application Developer for WebSphere Software (RAD) Versions 9.0 and 9.0.1 project files and workspaces (Java and IBM Worklight only) can be scanned - and the AppScan Source for Development (Eclipse plug-in) can be applied to RAD Versions 9.0 and 9.0.1.
  - You can now scan an Xcode project from the AppScan Source for Development Eclipse plug-in.
- **Windows and Linux**: Rational Application Developer for WebSphere Software (RAD) Versions 8.5.5 and 9.0.1 project files and workspaces (Java and IBM Worklight only) can be scanned - and the AppScan Source for Development (Eclipse plug-in) can be applied to RAD Versions 8.5.5 and 9.0.1.
- Eclipse Versions 4.3.1 and 4.3.2 project files and workspaces (Java and IBM Worklight only) can be scanned - and the AppScan Source for Development (Eclipse plug-in) can be applied to these versions of Eclipse.
- Rational Team Concert Versions 4.0.5 and 4.0.6 are now supported defect tracking systems.
- Xcode 5.0 for Objective-C (for iOS applications only) is now a supported compiler on OS X.

IBM Worklight integration

The AppScan Source for Development Eclipse plug-in now integrates with IBM Worklight. When AppScan Source for Development and IBM Worklight are installed to your Eclipse-based environment, you have the option to scan Worklight projects, applications, environments, and HTML files.

Using AppScan Source for Development without an AppScan Enterprise Server

As of AppScan Source Version 9.0, the AppScan Source for Development plug-ins can be used without AppScan Enterprise Server. In server mode, you connect to the server to run scans and access shared data, just in previous product versions. In the new local mode, AppScan Source for Development runs without ever connecting to a server - and you cannot access shared items such as filters, scan configurations, and custom rules.

Important: If you are using a floating license in local mode, you must still have a connection to the license server to be able to use AppScan Source for Development.

Optional AppScan Source for Development Eclipse plug-in quality component

As of AppScan Source Version 9.0, the AppScan Source for Development Eclipse plug-in quality component is provided as an optional installation.

Floating license option for AppScan Source for Automation

As of AppScan Source Version 9.0, AppScan Source for Automation has a floating license option.

Enhanced and new scanning support

- Performance is now improved when scanning JavaScript.
- Android KitKat (4.4) is now supported.
- AppScan Source now supports scanning applications that use these application programming interfaces (API): Worklight, Cordova, HTML5, JQuery, and JQuery Mobile.

Windows 7 machines that are configured to use the United States Government Configuration Baseline (USGCB)

AppScan Source supports scanning applications on Windows 7 machines that are configured with the USGCB specification.

Note: On machines that are configured with the USGCB specification, AppScan Source does not support defect tracking system integration with HP Quality Center or Rational ClearQuest.

Quality analysis feature deprecation in Version 9.0

The Java and C++ code quality analysis features are deprecated as of AppScan Source Version 9.0. These features can still be used in this version, but will not be supported or available in future versions.

Migrating to AppScan Source Version 9.0 from Version 8.7

This topic contains migration information for changes that went into AppScan Source Version 8.8. If you are upgrading AppScan Source Version 8.7 to Version 9.0, refer to this set of migration instructions in addition to the topic that describes migrating from Version 8.8 to Version 9.0.

- “Changes to findings classifications”
- “Default settings changes that will improve scan coverage” on page 7
- “Restoring AppScan Source predefined filters from previous versions” on page 8

Changes to findings classifications

As of AppScan Source Version 8.8, findings classifications have changed. This table lists the old classifications mapped to the new classifications:

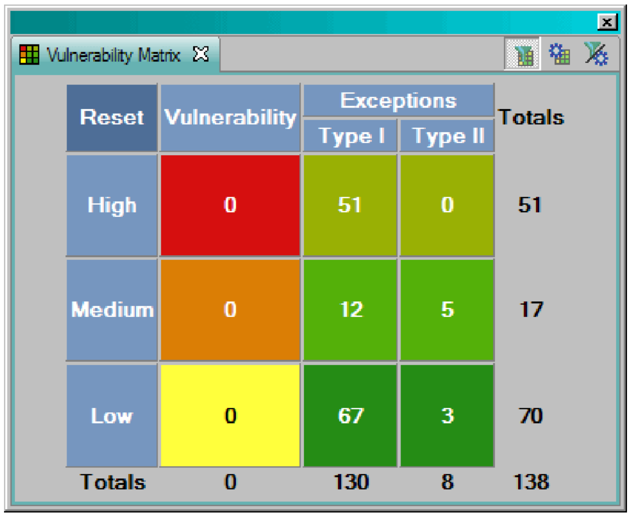
Table 1. Findings classification changes

Findings classifications prior to AppScan Source Version 8.8	Classifications in AppScan Source Version 8.8
Vulnerability	Definitive security finding

Table 1. Findings classification changes (continued)

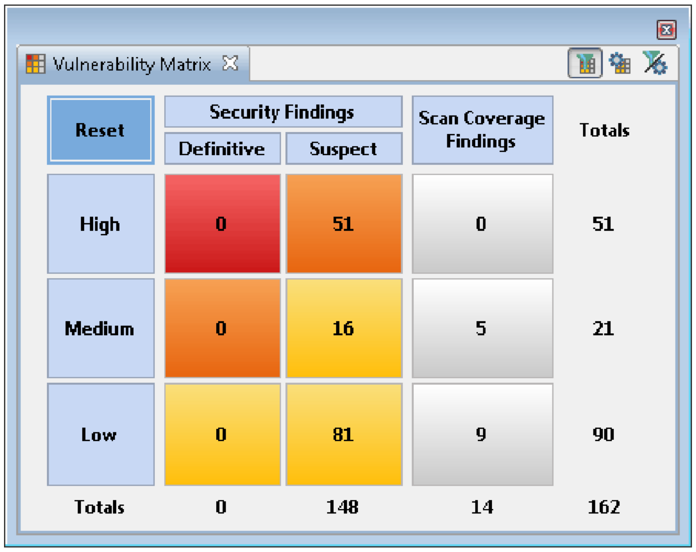
Findings classifications prior to AppScan Source Version 8.8	**Classifications in AppScan Source Version 8.8**
Type I Exception	Suspect security finding
Type II Exception	Scan coverage finding

An example of these changes can be seen in the Vulnerability Matrix view.



Reset	Vulnerability	Exceptions		Totals
		Type I	Type II	
High	0	51	0	51
Medium	0	12	5	17
Low	0	67	3	70
Totals	0	130	8	138

As of Version 8.8, the view looks like this:



Reset	Security Findings		Scan Coverage Findings	Totals
	Definitive	Suspect		
High	0	51	0	51
Medium	0	16	5	21
Low	0	81	9	90
Totals	0	148	14	162

Default settings changes that will improve scan coverage

As of AppScan Source Version 8.8:

- The default value of `show_informational_findings` in `scan.ozsettings` has changed from `true` to `false`.

- The default value of `wafl_globals_tracking` in `ipva.ozsettings` has changed from `false` to `true`. This setting enables AppScan Source to find dataflow between different components of a framework-based application (for example, dataflow from a controller to a view).

The change to `show_informational_findings` will result in assessments not including findings with a severity level of **Info** by default.

Note: If you have scan configurations that were created prior to Version 8.8 that did not explicitly set values for these settings, the scan configurations will now use their new default values.

Restoring AppScan Source predefined filters from previous versions

In AppScan Source Version 8.8, predefined filters were improved to provide better scan results. If you need to continue using the predefined filters from older versions of AppScan Source (archived filters are listed in “AppScan Source predefined filters (Version 8.7.x and earlier)” on page 111), follow the instructions in “Restoring archived predefined filters” on page 113.

AppScan Source for Analysis overview

AppScan Source for Analysis is a tool for analyzing code and providing specific information about source code vulnerabilities in critical systems. AppScan Source for Analysis lets you centrally manage your software risk across multiple applications, or even your entire portfolio. You can scan source code, triage, and eliminate vulnerabilities before they become a liability to your organization.

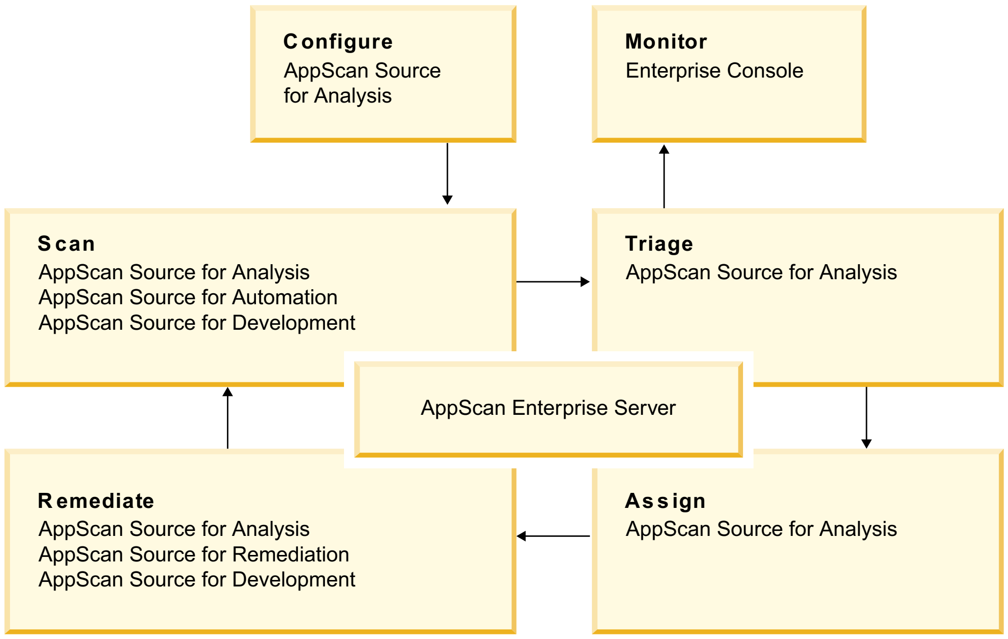
AppScan Source for Analysis provides audit and quality assurance teams with tools to scan source code, triage results, and submit flaws to defect tracking systems.

Armed with in-context intelligence from the AppScan Source Security Knowledgebase, analysts, auditors, managers, and developers can:
- Scan selected source code on-demand to locate critical vulnerabilities
- Receive precise remediation advice and invoke their preferred development environment and code editor directly from analysis
- Trace tainted data through a precise, interactive call graph from input to output
- Enforce coding policies, verifying approved input validation and encoding routines through AppScan Source trace
- Learn and implement secure programming best practices during software development

Workflow

After installation, deployment, and user management, the AppScan Source workflow consists of these basic steps.
1. **Set security requirements**: A manager or security expert defines vulnerabilities and how to judge criticality.
2. **Configure applications**: Organize applications and projects.
3. **Scan**: Run the analysis against the target application to identify vulnerabilities.

4. **Triage and analyze results**: Security-minded staff study results to prioritize remediation workflow and separate real vulnerabilities from potential ones, allowing triage on critical issues to begin immediately. Isolate the issues you need to fix first.
5. **Customize the Knowledgebase**: Customize the AppScan Source Security Knowledgebase to address internal policies.
6. **Publish scan results**: Add scan results to the AppScan Source Database or publish them to the AppScan Enterprise Console.
7. **Assign remediation tasks**: Assign defects to the development team to resolve vulnerabilities.
8. **Resolve issues**: Eliminate vulnerabilities by rewriting code, removing flaws, or adding security functions.
9. **Verify fixes**: The code is scanned again to assure that vulnerabilities are eliminated.


Configure
AppScan Source for Analysis
Monitor
Enterprise Console
Scan
AppScan Source for Analysis
AppScan Source for Automation
AppScan Source for Development
Triage
AppScan Source for Analysis
AppScan Enterprise Server
Remediate
AppScan Source for Analysis
AppScan Source for Remediation
AppScan Source for Development
Assign
AppScan Source for Analysis

Important concepts

Before you begin to use or administer AppScan Source, you should become familiar with fundamental AppScan Source concepts. This section defines basic AppScan Source terminology and concepts. Subsequent chapters repeat these definitions to help you understand their context in AppScan Source for Analysis.

AppScan Source for Analysis *scans* source code for vulnerabilities and produces *findings*. Findings are the vulnerabilities identified during a scan, and the result of a scan is an *assessment*. A *bundle* is a named collection of individual findings and is stored with an application.

Applications, their attributes, and projects are created and organized in AppScan Source for Analysis:

- **Applications**: An application contains one or more projects and their related attributes.

- **Projects**: A project consists of a set of files (including source code) and their related information (such as configuration data). A project is always part of an application.
- **Attributes**: An attribute is a characteristic of an application that helps organize the scan results into meaningful groupings, such as by department or project leader. You define attributes in AppScan Source for Analysis.

The principal activity of AppScan Source for Analysis is to scan source code and analyze vulnerabilities. Assessments provide an analysis of source code for vulnerabilities including:

- **Severity**: High, medium, or low, indicating the level of risk
- **Vulnerability Type**: Vulnerability category, such as SQL Injection or Buffer Overflow
- **File**: Code file in which the finding exists
- **API/Source**: The vulnerable call, showing the API and the arguments passed to it
- **Method**: Function or method from which the vulnerable call is made
- **Location**: Line and column number in the code file that contains the vulnerable API
- **Classification**: Security finding or scan coverage finding. For more information, see "Classifications."

Classifications

Findings are classified by AppScan Source to indicate whether they are security or scan coverage findings. Security findings represent actual or likely security vulnerabilities - whereas scan coverage findings represent areas where configuration could be improved to provide better scan coverage.

Each finding falls into one of these *classifications*:

- **Definitive** security finding: A finding that contains a definitive design, implementation, or policy violation that presents an opportunity for an attacker to cause the application to operate in an unintended fashion.

  This attack could result in unauthorized access, theft, or corruption of data, systems, or resources. Every definitive security finding is fully articulated, and the specific underlying pattern of the vulnerable condition is known and described.
- **Suspect** security finding: A finding that indicates a suspicious and potentially vulnerable condition that requires additional information or investigation. A code element or structure that can create a vulnerability when used incorrectly.

  A suspect finding differs from a definitive finding because there is some unknown condition that prevents a conclusive determination of vulnerability. Examples of this uncertainty can be the use of dynamic elements, or of library functions for which the source code is not available. As a result, there is an additional level of research that is required to confirm or reject a suspect finding as definitive.
- **Scan coverage** finding: Findings that represent areas where configuration could be improved to provide better scan coverage (for example, lost sink findings).

Note: In some cases, a classification of **None** may be used to denote a classification that is neither a security finding nor a scan coverage finding.

Logging in to AppScan Enterprise Server from AppScan Source products

Most AppScan Source products and components require a connection to an AppScan Enterprise Server. The server provides centralized user management capabilities and a mechanism for sharing assessments via the AppScan Source Database.

When you launch AppScan Source for Analysis, you are prompted to log in. If you are running AppScan Source for Development in server mode, you are prompted to log in when you first initiate an action that needs access to the server, such as launching a scan, viewing scan configurations, or changing your password.

In AppScan Source for Analysis, when logging in, you are prompted for:

- **User ID**: Specify your user ID (depending on how your account was set up, this is a user ID that exists both on the AppScan Enterprise Server and in the AppScan Source Database - or it is a user ID that exists only in the AppScan Source Database).
- **Password**: Specify the password for your user ID.
- **AppScan Enterprise Server**: Specify the URL for your AppScan Enterprise Server instance.

In AppScan Source for Development, when logging in, you are prompted for:

- **Server URL**: Specify the URL for your AppScan Enterprise Server instance.
- **User ID**: Specify your user ID (depending on how your account was set up, this is a user ID that exists both on the AppScan Enterprise Server and in the AppScan Source Database - or it is a user ID that exists only in the AppScan Source Database).
- **Password**: Specify the password for your user ID.

Login actions are also required when running AppScan Source for Automation or the AppScan Source command line interface (CLI). See the *IBM Security AppScan Source Utilities User Guide* for more information.

To learn about AppScan Enterprise Server SSL certificates, see "AppScan Enterprise Server SSL certificates" on page 12.

Changing your password

This topic describes the steps for changing your password. If your AppScan Enterprise Server is configured to use LDAP authentication or Windows authentication, this functionality is not available.

Procedure

1. Choose **Admin** > **Change Password** from the main menu.
2. Enter your old password.
3. Type and confirm a new password.
4. Click **OK** to change the password.

   Note: The credentials used by AppScan Enterprise Server users are always the same credentials that are used to log in to AppScan Source. If the credentials are changed in either product, the change will automatically be in effect in the other product.

AppScan Enterprise Server SSL certificates

When the AppScan Enterprise Server is installed, it should be configured to use a valid SSL certificate. If this is not done, you will receive an untrusted connection message when logging in to the server from AppScan Source for Analysis or the AppScan Source command line interface (CLI) - or AppScan Source for Development on Windows and Linux.

SSL certificate storage location

Certificates that have been permanently accepted are stored in `<data_dir>\config\cacertspersonal` and `<data_dir>\config\cacertspersonal.pem` (where `<data_dir>` is the location of your AppScan Source program data, as described in "Installation and user data file locations" on page 258). Remove these two files if you no longer want the certificates permanently stored.

AppScan Source for Automation and SSL certificate validation

By default, certificates are automatically accepted when using AppScan Source for Automation. This behavior is determined by the `ounceautod_accept_ssl` setting in the Automation Server configuration file (`<data_dir>\config\ounceautod.ozsettings` (where `<data_dir>` is the location of your AppScan Source program data, as described in "Installation and user data file locations" on page 258)). If this setting is edited so that `value="true"` is set to `value="false"`, SSL validation will be attempted and logging in or publishing to AppScan Enterprise Console will fail with error if an invalid certificate is encountered.

AppScan Source command line interface (CLI) and SSL certificate validation

By default, when using the CLI `login` command, SSL validation will be attempted and logging in or publishing to AppScan Enterprise Console will fail with error if an invalid certificate is encountered (if you have not already permanently accepted the certificate while logging in via another AppScan Source client product). This behavior can be modified by using the option `-acceptssl` parameter when issuing the `login` command. When this parameter is used, SSL certificates are automatically accepted.

AppScan Source and accessibility

Accessibility affects users with physical disabilities, such as restricted mobility or limited vision. Accessibility issues can impede the ability to use software products successfully. This topic outlines known AppScan Source accessibility issues and their workarounds.

Using JAWS Screen Reading Software with the AppScan Source installer

To use Freedom Scientific JAWS (http://www.freedomscientific.com/products/fs/jaws-product-page.asp) when running the AppScan Source installer, you must install Java Access Bridge in the AppScan Source JVM. This will allow JAWS to properly speak labels and controls in the installer panels.

- Information about the Java Access Bridge (including the download link and installation instructions) can be found at http://www.oracle.com/technetwork/java/javase/tech/index-jsp-136191.html.

- Information about the InstallAnywhere requirement for installing the Java Access Bridge can be found at http://kb.flexerasoftware.com/selfservice/documentLink.do?externalID=Q200311.

Using JAWS Screen Reading Software in user interface panels with descriptive text

Many parts of the AppScan Source user interface contain descriptive text. In most cases, you must use the JAWS `Insert+B` keystroke to be able to read this descriptive text.

Legal notices for IBM Security AppScan Source V9.0 documentation

Copyright

Chapter 2. Configuring applications and projects

Before you scan, you must configure applications and projects. This section explains the Application Discovery Assistant, New Application Wizard, and the New Project Wizard. You will learn how to configure attributes for AppScan Source for Analysis. In addition, this section teaches you how to add existing applications and projects for scanning - and how to add files to projects.

AppScan Source for Analysis configuration includes application creation, source code configuration, and attribute configuration. After you configure and scan, you proceed to triage. You can configure your source code in the Properties view or with the New Project Wizard. This chapter guides you through the Wizard. See "Properties view" on page 223 for an overview of application and project properties.

AppScan Source for Analysis uses an application/project model that directly imports Microsoft Visual Studio, Eclipse, Rational Application Developer for WebSphere Software (RAD), or AppScan Source projects previously created with the AppScan Source utilities (refer to the *IBM Security AppScan Source Utilities User Guide* for further details).

You can add and configure projects of various types and containing a variety of languages - specifying settings gathered from the target code base and its build procedures. During the configuration, you can specify directories and files to exclude from a scan.

Before you scan, you must configure *applications* and *projects*. An application is a container for projects; a project is the set of files to scan and the settings (configuration) used.

AppScan Source application and project files

AppScan Source applications and projects have corresponding files that maintain configuration information required for scanning, as well as triage customization. It is recommended that these files reside in the same directory as the source code, since configuration information (dependencies, compiler options, and so forth) required to build the projects is very similar to that required for AppScan Source to scan them successfully. Best practice includes managing these files with your source control system.

When you use supported build integration tools (for example, Ounce/Ant or Ounce/Maven) to generate AppScan Source applications and project files, it is recommended that you update these files in source control as part of your build automation, to facilitate sharing them across the development team. When a developer updates the local view of the files in source control, the AppScan Source application and project files update as well. This ensures that the entire team is working with a consistent set of files.

Applications and projects created in AppScan Source for Analysis have a `.paf` and `.ppf` extension respectively. These files are generated when you manually create and configure an application or project in the AppScan Source for Analysis user interface or via supported AppScan Source utilities.

On Windows, When you import Microsoft solutions and projects into AppScan Source for Analysis, files with `.gaf` and `.gpf` extensions are created for them.

On OS X, When you import Xcode directories and projects into AppScan Source for Analysis, files with `.xcodeproj.gaf` and `.xcodeproj.gpf` extensions are created for them.

Note: When an Eclipse Importer runs on an Eclipse or Rational Application Developer for WebSphere Software (RAD) workspace, AppScan Source creates intermediate files with `.ewf` and `.epf` extensions. These files are required for the initial import into AppScan Source for Analysis and for future scans.

Table 2. AppScan Source files

AppScan Source File Extension	Description
`ppf`	• AppScan Source project file • Generated when you create a project with AppScan Source for Analysis or supported AppScan Source utilities • User-named
`paf`	• AppScan Source application file • Generated when you create an application with AppScan Source for Analysis or supported AppScan Source utilities • User-named

`gaf`	• AppScan Source application file that is generated when you import Microsoft solutions
	• Used to hold custom application information such as exclusions and bundles
	• Adopts the name of the imported workspace or solution. For example:
	`d:\my_apps\myapp.sln`
	`d:\my_apps\myapp.sln.gaf`

`gpf`	• AppScan Source project file that is generated when you import Microsoft projects
	• Used to hold custom project information such patterns and exclusions
	• Adopts the name of the imported project: For example:
	`d:\my_projects\myproject.vcproj`
	`d:\my_projects\myproject.vcproj.gpf`

Table 2. AppScan Source files (continued)

AppScan Source File Extension	Description

`.xcodeproj.gaf`	• AppScan Source application file that is generated when you import Xcode directories
	• Used to hold custom application information such as exclusions and bundles
	• Adopts the name of the imported workspace or solution. For example:
	`/Users/myUser/myProject.xcodeproj`
	`/Users/myUser/myProject.xcodeproj.gaf`

`.xcodeproj.gpf`	• AppScan Source project file that is generated when you import Xcode projects
	• Used to hold custom project information such patterns and exclusions
	• Adopts the name of the imported project: For example:
	`/Users/myUser/myProject.xcodeproj`
	`/Users/myUser/myProject.xcodeproj.gpf`

`ewf`	• Eclipse workspace file • Produced when you import an Eclipse workspace into AppScan Source • The Eclipse exporter creates the file based on information in the Eclipse workspace - AppScan Source then imports the file
`epf`	• Eclipse project file • Produced when an Eclipse project is imported into AppScan Source • The Eclipse exporter creates the file based on information in the Eclipse project - AppScan Source then imports the file

Configuring applications

You can use the New Application Wizard or the Application Discovery Assistant to create applications. The Application Discovery Assistant automates application setup for you, whereas the New Application Wizard allows you to add applications, guiding you through the configuration process. The wizard helps you manually create a project or add existing projects to an application. This section describes these two methods for adding application and basic configuration tasks.

Note: The Application Discovery Assistant quickly creates and configures applications and projects for Java source code and Microsoft Visual Studio solutions - or Eclipse or IBM Rational Application Developer for WebSphere Software (RAD) workspaces that contain Java projects. To create an application for any other supported language, use the New Application Wizard - or import supported applications to AppScan Source for Analysis.

You must create a new application (see "Creating a new application with the New Application Wizard" or "Using the Application Discovery Assistant to create applications and projects" on page 21) or add an existing application (see "Adding an existing application" on page 24) before adding projects. If you use Microsoft Visual Studio, you already arrange your source files in projects. AppScan Source for Analysis allows you to import Solutions and treat them as AppScan Source applications.

The following table lists the application file types that you can open and scan with AppScan Source for Analysis.

Table 3. Supported Application File Types

Application	**File type**
• Microsoft Visual Studio 8.0 • Microsoft Visual Studio 9.0 • Microsoft Visual Studio 10.0	`.sln` (Solution)
• Eclipse workspace (Java only) • RAD workspace (Java only) See the AppScan Source system requirements to learn which versions of Eclipse and RAD are supported for workspace scanning.	`<workspace directory>` or `.ewf` The workspace directory contains an additional directory, `.metadata`.
AppScan Source application file	`.paf`

Tip: An icon appears in the Explorer view to indicate an imported application (see "Application and project indicators" on page 62).

Note: When applications and projects are created using the New Application Wizard and New Project wizard, their file name is automatically assigned according to the **Name** entered in the wizard (for example, if a project is being created and **MyProject** is entered in the **Name** field, the project filename will be `MyProject.ppf`). Application and project names can be renamed using the Properties view.

Creating a new application with the New Application Wizard

Procedure

1. Complete one of these actions:
   - Select **File** > **Add Application** > **Create a new application** from the main menu bar.
   - In the Explorer view toolbar, click the **Add Application Menu** down-arrow button and select **Create a new application** from the menu.
   - In the Explorer view, right-click **All Applications** and then select **Add Application** > **Create a new application** from the menu.
2. Enter a **Name** for the application.
3. Browse to the **Working Directory** in which to save the application. The new application file name extension will be `.paf`.

4. Click **Next** to configure the projects comprising the application or **Finish** to add the application without configuring any projects. Help for configuring and adding projects is provided later in this section.

Using the Application Discovery Assistant to create applications and projects

AppScan Source includes a powerful Application Discovery Assistant which allows you to quickly create and configure applications and projects for Java source code and Microsoft Visual Studio solutions. The Application Discovery Assistant also allows you to locate Eclipse or Rational Application Developer for WebSphere Software (RAD) workspaces that contain Java projects. The Application Discovery Assistant allows you to point to your source, solution, or workspace directory - and then AppScan Source handles the rest.

About this task

You can use the Application Discovery Assistant to search a location that contains a combination of Java source, Microsoft Visual Studio solutions, and/or Eclipse workspaces. The final panel of the Application Discovery Assistant allows you to specify application/project structure preferences for Java only. This panel has no bearing on the placement of application and project files for Microsoft Visual Studio solution or Eclipse workspaces - where application files are automatically placed in the root of the solution or workspace - and project files are automatically placed in the root of individual solution or workspace projects.

Procedure

1. Complete one of these actions to launch the Application Discovery Assistant:
   - Select **File** > **Add Application** > **Discover Applications** from the main menu bar.
   - In the Explorer view **Quick Start** section, select **Discover Applications**.
   - In the Explorer view toolbar, click the **Add application menu** down-arrow button and select **Discover Applications** from the menu.
   - In the Explorer view, right-click **All Applications** and then select **Add Application** > **Discover Applications** from the menu.
2. In the Search Location panel, specify the location that contains the source code, solutions, or workspaces that you want to scan. In addition, you can set the scan to begin immediately after completing application discovery.

   From here, you can click **Next** to set additional Application Discovery Assistant options (such as external dependency specification, exclusion rules, and Java application/project structure preferences) - or you can click **Start** to begin application discovery. If you click **Start**:
   - No external dependency locations will be set. If your application has external dependencies and they are not specified, scan results will be negatively impacted.
   - Out-of-the box exclusion rules will be used (see "Default Application Discovery Assistant exclusion rules" on page 24 for a list of the default rules).
   - If you are locating Java source, one project and application will be created (the single project will contain all source roots that are found).

   If you click **Next**, proceed to the next step.

3. In the External Dependencies panel, set a path for each external dependency that your application has (for example, a path to a JDK or web server). To complete this panel, follow these instructions:
   a. To add an external dependency, click inside the table or click **Add** - and then type in or browse for the external dependency path. To accept a path that you have entered by keyboard, press the keyboard Enter key.

      Tip: Typing into the dependency path field while it is being edited causes directories to be listed that you can select. You must at least type in a drive letter. For the path that is specified, all folders that it contains will be listed.
   b. To remove an external dependency path, select it and click **Delete**.
   c. To modify an external dependency path, click inside the path and then type in or browse for the external dependency path.

   From here, you can click **Next** to set additional Application Discovery Assistant options - or you can click **Start** to begin application discovery. If you click **Start**:
   - Out-of-the box exclusion rules will be used (see "Default Application Discovery Assistant exclusion rules" on page 24 for a list of the default rules).
   - If you are locating Java source, one project and application will be created (the single project will contain all source roots that are found).

   If you click **Next**, proceed to the next step.
4. In the Exclusion Rules panel, specify rules for filtering out files and directories. Rules are set by PERL, Grep, EGrep, or exact match regular expression. For example, if you want to exclude a directory named `temp` from the Application Discovery search, you could add a PERL `.*[\\/]temp` exclusion rule.

   By default, a set of PERL regular expressions are provided for excluding some common directories (see "Default Application Discovery Assistant exclusion rules" on page 24 for the complete list). To modify this list or create new rules, follow these instructions:
   a. To modify an existing exclusion rule, click inside the rule to activate the rule editor. Once you are finished editing the rule, click away from it or press the keyboard **Enter** key.

      To modify the regular expression type of an existing rule, click inside the **Regex Type** cell of the rule and then select the regular expression type from the menu.
   b. To add an exclusion rule, click **Add**. This adds a new rule to the table, which you can alter by following the above instructions for modifying rules.
   c. To remove an exclusion rule, select it and click **Delete** (or click **Delete All** to remove all exclusion rules currently listed in the panel).

   Important: Valid exclusion rules are denoted by check mark in the table - and invalid rules are denoted by a red `X`. You will not be able to start Application Discovery or continue in the Application Discovery Assistant until all rules are valid.

   From here:
   - If you are searching for Java source only, you can click **Next** to set Application Discovery Assistant application/project structure preferences - or you can click **Start** to run the assistant.
   - If you are only searching for Microsoft Visual Studio solutions or Eclipse workspaces, click **Start** to run the assistant. Clicking **Next** will cause the assistant to proceed to a panel that applies only to Java source discovery.

If you click **Next**, proceed to the next step.

5. The Application and Project Creation panel applies only to Java source discovery. In it, specify the structure of the applications and projects that will be created:
   a. To create a single project for all source roots that are found, select **Create a single project** in the **Projects** menu. With this selection, you will only have the option of creating a single application.
   b. To create a separate project for each source root that is found, select **Create a project for each source root found** in the **Projects** menu. With this selection, you can choose to create one application or multiple applications. To create a single application that contains all projects that are created, select **Create a single application** in the **Applications** menu. To create an application for each project that is created, select **Create an application per project** in the **Applications** menu.

   In addition, choose a location to store the application and project definition files.

   If you choose **Organize the files for me**:
   - If you are creating a single project, the project and application files will be created in the search location.
   - If you are creating a project for each source root in a single application, the project file for each source root will be created in the directory above the source root - and the application file will be created in the search location.
   - If you are creating a project for each source root and an application for each project, the project and application files for each source root will be created in the directory above the source root.

   If you specify a directory, all application and project files will be created in that directory.
6. If you want to change any of the settings made in previous panels, click **Back**. When you are satisfied with the Application Discovery settings, click **Start** to scan the search location for source roots.

Results

When Application Discovery is complete, new applications and projects that were created as a result of Application Discovery appear in the Explorer view, ready for scanning (if you set the scan to begin immediately after completing application discovery, the scan will begin).

If problems were encountered during discovery, the Application Discovery Assistant provides a discovery report upon completion. For example, if your application has external dependencies that were not specified in the External Dependencies panel, the report will contain warnings indicating that external dependencies cannot be resolved. In the discovery report:
- Click **Finish** to create the applications and projects. If **Ignore warnings and scan anyway** is selected, the applications and projects will be scanned immediately.
- Click **Back** to alter Application Discovery Assistant settings or run Application Discovery again.
- Click **Cancel** to close the discovery report without creating applications or projects.

Default Application Discovery Assistant exclusion rules

When using the Application Discovery Assistant, default exclusion rules will be used if the Exclusion Rules panel is not modified - or if you start the Application Discovery after specifying the search directory. Default Application Discovery exclusion rules are listed in this topic.

Table 4. Default Application Discovery exclusion rules

Exclusion rule	Regular expression type
`.*[\\/]example`	PERL
`.*[\\/]test`	PERL
`.*[\\/]demo`	PERL
`.*[\\/]sample`	PERL

Adding an existing application

Existing applications can be added for scanning by dragging and dropping them into the Explorer view - or by using the **Add Application** action.

To learn how to add an existing application, see these topics:
- "Adding an existing application with user interface actions"
- "Adding an existing application with drag and drop"

Adding an existing application with user interface actions

Procedure

1. Complete one of these actions:
   - Select **File** > **Add Application** > **Open an existing application** from the main workbench menu.
   - In the Explorer view toolbar, click the **Add Application Menu** down-arrow button and select **Open an existing application** from the menu.
   - In the Explorer view, right-click **All Applications** and then select **Add Application** > **Open an existing application** from the menu.
2. Select the directory that contains the saved application file (`.paf`, `.sln`, `.dsw`, or `.ewf`).
3. Open the application file.

Adding an existing application with drag and drop

Procedure

1. On your workstation, locate the application (`.paf`, `.sln`, `.dsw`, or `.ewf`) that you want to add for scanning.

   Note: You cannot drag and drop workspace directories.
2. Select the application and then drag it to the Explorer view.
3. Drop the selection on or beneath the **All Applications** node.

Adding multiple applications

Rather than adding just one application at a time, when you first begin working with AppScan Source for Analysis, you may want to import multiple applications. The Select Applications dialog box allows you to select a root directory from which to search for AppScan Source applications (`.paf`) or Microsoft Solution files (`.sln`). Multiple applications can also be added for scanning by dragging and dropping them into the Explorer view.

To learn how to add an multiple applications, see these topics:
- "Adding multiple applications with user interface actions"
- "Adding multiple applications with drag and drop"

Adding multiple applications with user interface actions

Procedure

1. Select **File** > **Add Application** > **Multiple Applications** from the main workbench menu.
2. In the Select Applications dialog box, browse to the root directory that contains the applications that you want to import. Select the **Recurse into subdirectories** check box to search in subdirectories.
3. Complete one of these actions:
   - Click **Finish** to import the applications and add them to the Explorer view.
   - Click **Next** to view the search results and select the applications to import. Then click **Finish**.

Adding multiple applications with drag and drop

Procedure

1. On your workstation, locate the applications (`.paf`, `.sln`, `.dsw`, or `.ewf` files) that you want to add for scanning.

   Note: You cannot drag and drop workspace directories.
2. Select or multiselect the applications and then drag them to the Explorer view.
3. Drop the selection on or beneath the **All Applications** node.

Adding an Eclipse or Eclipse-based product workspace

If you have an Eclipse or Rational Application Developer for WebSphere Software (RAD) workspace that contains Java and/or IBM Worklight projects, you can import it to AppScan Source for Analysis.

Before you begin

Before you add the workspace, be certain that you have installed and updated the development environment as described in "Configuring your development environment for Eclipse and Rational Application Developer for WebSphere Software (RAD) projects" on page 26.

Procedure

1. Complete one of these actions:
   - Select **File** > **Add Application** > **Import an existing Eclipse/RAD workspace** from the main workbench menu.
   - In the Explorer view toolbar, click the **Add application menu** down-arrow button and select **Import an existing Eclipse/RAD workspace** from the menu.
   - In the Explorer view, right-click **All Applications** and then select **Add Application** > **Import an existing Eclipse/RAD workspace** from the menu.
2. Select the **Workspace Type**.

3. Browse to the workspace, select the directory, and then click **OK** to add the workspace.

Configuring your development environment for Eclipse and Rational Application Developer for WebSphere Software (RAD) projects

Before you import an Eclipse or Rational Application Developer for WebSphere Software (RAD) project, you must properly configure the development environment. Although Eclipse is the basis for each project type, AppScan Source distinguishes between the different versions. AppScan Source supports these external Eclipse environments: Eclipse Versions 3.6, 3.7, 3.8, 4.2, 4.2.x, 4.3, 4.3.1, and 4.3.2 - and Rational Application Developer for WebSphere Software (RAD) Versions 8.0.x, 8.5, 8.5.1, 8.5.5, 9.0, and 9.0.1.

To learn more about configuring your development environment for this, see these help topics:
- "Eclipse or Application Developer updates"
- "Eclipse workspace importers: Eclipse or Rational Application Developer for WebSphere Software (RAD) preference configuration"

Eclipse or Application Developer updates

For Eclipse or Application Developer environments external to AppScan Source, you must make sure that you have the appropriate software updates installed. These instructions explain how to obtain and install the updates. The procedure may vary for different versions.

Before you begin

Important: AppScan Source for Development requires a Java Runtime Environment (JRE) that is Version 1.5 or higher. If your environment points to a JRE that does not meet this requirement, edit the `eclipse.ini` file in the Eclipse installation directory so that it points to a JRE that does meet this requirement. For information about making this change to the `eclipse.ini` file, see the *Specifying the JVM* section of http://wiki.eclipse.org/Eclipse.ini.

Procedure

1. On the Eclipse **Help** menu, select the option to install new software (the menu label varies, depending on the version of Eclipse that you are using).
2. Select the option to add a Local Update Site.
3. When prompted for the location of the site, navigate to the AppScan Source installation directory.
4. Add this update site and follow the displayed steps until prompted to restart Eclipse.
5. The AppScan Source menu appears after the installation completes.

Eclipse workspace importers: Eclipse or Rational Application Developer for WebSphere Software (RAD) preference configuration

The AppScan Source for Analysis installation provides a default Eclipse importer. This importer identifies the location of Eclipse and the JRE. If the default Eclipse importer is unable to import your workspace, it may be necessary to create a new Eclipse importer.

Before you begin

Each importer configuration represents an installation of Eclipse or Rational Application Developer for WebSphere Software (RAD). To use these configurations to import existing workspaces and projects to AppScan Source for Analysis, you may also need to install the AppScan Source for Development plug-ins into the Eclipse environment.

Before adding a RAD workspace, you must create a configuration for the workspace type.

Procedure

1. In AppScan Source for Analysis, select **Edit** > **Preferences** from the main workbench menu.
2. Select **Eclipse Workspace Importers**.
3. Click **Create a new configuration** and then complete the New Import Configuration dialog box to create a new configuration:
   - **Product**: Select the appropriate product

     Note: If the product that was used to create the workspace is not available for selection, ensure that you have completed the configuration steps outlined in “Eclipse or Application Developer updates” on page 26 before attempting to create the workspace importer.
   - **Name**: Importer name
   - **Location**: Path to the base directory of the Eclipse installation
   - **JRE Location**: Path to the root directory of the Java Runtime Environment (JRE). Use a JDK in `<install_dir>\JDKS` (where `<install_dir>` is the location of your AppScan Source installation) or any other preferred JDK.
4. Click **OK**.
5. To identify the importer as default, select it and click **Make the selected configuration the default**. This causes an icon to display in the importer's **Default** column.

Creating a new project for an application

After you add an application, you add projects to it. Project types that can be scanned include: Java/JSP, ASP, C/C++, COBOL, ColdFusion, Xcode, Java/JSP, .NET Assembly, Pattern Based, Perl, PHP, PL/SQL, T-SQL, Visual Basic, and JavaScript.

About this task

If you use `make` to compile your project, it is recommended that you use the Ounce/Make utility to create a project file and then add the project file. If you use `ant` to compile your project, use Ounce/Ant to create a project file and then add the project file. Refer to the *IBM Rational AppScan Source Edition Utilities User Guide* for details about Ounce/Make and Ounce/Ant.

Note: The default file encoding for AppScan Source projects is **ISO-8859-1**. The default file encoding can be changed in the General preference page.

Note: When applications and projects are created using the New Application Wizard and New Project wizard, their file name is automatically assigned according to the **Name** entered in the wizard (for example, if a project is being created and **MyProject** is entered in the **Name** field, the project filename will be `MyProject.ppf`). Application and project names can be renamed using the Properties view.

Procedure

1. In the Explorer view, select the application that you want to add the project to (if you have not already added an application, see "Configuring applications" on page 19).
2. Complete one of these actions to open the New Project Wizard:
   a. Select **File** > **Add Project** > **New Project** from the main workbench menu.
   b. Right-click the selected application and choose **Add Project** > **New Project** from the context menu.
3. Complete the New Project Wizard.

Adding an existing project

You can add AppScan Source projects (`.ppf` files) previously created with AppScan Source for Analysis to AppScan Source applications. You can also add Eclipse project files (`.epf`), projects created by any of the supported build integration tools (for example, Ounce/Maven or Ounce/Ant), or project files created with Microsoft Visual C/C++ (`.vcproj` or`.dsp`), VB.NET (`.vbproj`), or C# (`.csproj`).

This table lists the project file types that you can open and scan with AppScan Source for Analysis:

Table 5. Project File Types to Open

Project file type	**File extension**
Microsoft Visual Studio (version 6)	`.dsp`
Microsoft Visual Studio C/C++	`.vcproj`
Microsoft Visual Studio C#	`.csproj`
Microsoft Visual Studio Visual Basic	`.vbproj`
AppScan Source project file	`.ppf`
Eclipse project file	`.epf`

To learn how to add an existing project, see these topics:
- "Adding an existing project with user interface actions"
- "Adding an existing project with drag and drop" on page 29

Note: When you import an existing .NET project, you can specify additional assemblies to scan. Add these assemblies in the project's Properties view Additional Assemblies tab. When you add additional assemblies, you can combine .NET projects that build with assemblies (including third party assemblies) that do not build, in a single scan.

Adding an existing project with user interface actions

Procedure

1. In the Explorer view, select the application that you want to add the project to (if you have not already added an application, see "Configuring applications" on page 19).
2. Complete one of these actions:
   - Select **File** > **Add Project** > **Existing Project** from the main workbench menu.

- Right-click the selected application and choose **Add Project** > **Existing Project** from the context menu.

3. Browse to the project file to add it to the application.

Adding an existing project with drag and drop

Procedure

1. On your workstation, locate the project (`.ppf`, `.vcproj`, `.dsp`, `.vbproj`, `.csproj`, or `.war` file) that you want to add for scanning.

   Note: You cannot drag and drop files created by any of the supported build integration tools (for example, Ounce/Maven or Ounce/Ant).
2. Select the project and then drag it to the AppScan Source for Analysis Explorer view.
3. Complete one of these steps:
   a. Drop the selection in an existing application.
   b. Drop the selection on or beneath the **All Applications** node. Since projects must be contained by an application and this action does not add the project to an existing application, you will be prompted by the New Application Wizard to create a new application for the project. Enter a **Name** for the application and then browse to the **Working Directory** in which to save the application. Click **Finish** to create the new application (in the Explorer view, the added project will be contained within it).

Adding multiple projects

When you add multiple projects to an application, you can drag and drop them to the Explorer View - or you can browse a directory for projects and import some or all projects to the current application.

To learn how to add multiple projects, see these topics:
- "Adding multiple projects with user interface actions"
- "Adding multiple projects with drag and drop" on page 30

Adding multiple projects with user interface actions

Multiple projects can be added to an application from a directory (including subdirectories), an Eclipse or Rational Application Developer for WebSphere Software (RAD) workspace - or from a Microsoft solution file.

Procedure

1. In the Explorer view, select the application that you want to add the projects to (if you have not already added an application, see "Configuring applications" on page 19).
2. Complete one of these actions:
   - Select **File** > **Add Project** > **Multiple Projects** from the main workbench menu.
   - Right-click the selected application and choose **Add Project** > **Multiple Projects** from the context menu.
3. In the Add Multiple Projects dialog box, complete one of these actions:
   - Select **Import from Directory** and then browse to the root directory that contains the projects that you want to add. Select the **Recurse into subdirectories** check box to search in subdirectories.

- Select **Import from Eclipse/RAD Workspace**. Select the **Workspace Type** and then browse to the workspace. Select the workspace directory and then click **OK**.
- Select **Import from a Microsoft Solution File**. Browse to the file and select it - and then click **OK.**

4. Complete one of these actions:
   - Click **Finish** to add the projects to the application.
   - Click **Next** to view the search results and select the projects to add. Then click **Finish**.

Adding multiple projects with drag and drop

Procedure

1. On your workstation, locate the projects (`.ppf`, `.vcproj`, `.dsp`, `.vbproj`, or `.csproj`) that you want to add for scanning.

   Note: You cannot drag and drop files created by any of the supported build integration tools (for example, Ounce/Maven or Ounce/Ant).
2. Select or multiselect the projects and then drag them to the Explorer view.
3. Drop the selection in an existing application.

   Note: You can also drop the selection on or beneath the **All Applications** node, however this is not recommended. Rather, it is recommended that multiple projects be dropped into an existing application, or individually, if new applications are required.

   Since projects must be contained by an application and dropping projects on or beneath the **All Applications** node does not add the project to an existing application, you will be prompted by the New Application Wizard to create a new application for each project that you are adding to the view.

   To add multiple projects to a new application that does not yet exist, create the application first and then drag and drop the selected projects to it.

Adding a new ASP project

The Project Configuration Wizard helps you manually create an ASP project and add it to an application.

About this task

The steps in this topic direct you to complete all pages in the New Project Wizard (or New Application Wizard, if you are creating the project in it). However, some of the pages in the wizard are optional (required settings are complete when the **Finish** button is activated). Settings made in the wizard can be modified after project creation in the Properties view for a selected project. If you complete the New Project Wizard without completing optional pages, you can change the settings from those pages later on in the Properties view.

Note: For PHP, VB6, and Classic ASP, only ISO-8859-1 (Western Europe), UTF-8, and UTF-16 character sets are supported.

Procedure

1. In the Explorer view, select the application that you want to add the project to (if you have not already added an application, see “Configuring applications” on page 19).

2. Complete one of these actions to open the New Project Wizard:
   a. Select **File** > **Add Project** > **New Project** from the main workbench menu.
   b. Right-click the selected application and choose **Add Project** > **New Project** from the context menu.
3. In the Select Project Type page of the wizard, select **ASP** as the project type and then click **Next** to advance to the next wizard page.
4. In the Project Sources wizard page:
   a. Identify the project sources. Project sources consist of the directories in which you find project files, and any additional individual files to include in the project.

      Name the project and specify the working directory. The **Working Directory** is the location in which the AppScan Source project file (`.ppf`) will reside. It is also the base for all relative paths.
   b. Click **Add Source Root** to specify a source code root and the directories or files to include or exclude from the scan. After adding the source root, you can exclude certain directories or files from it. To do this, select the directory or file (or multiselect these items) in the source root, right-click the selection, and then choose **Exclude** from the menu. If you include or exclude files, the icon to the left of the file name changes.
5. Click **Next** to advance to the next wizard page.
6. In the ASP Project Configuration page:
   a. Configure the ASP project by identifying the ASP content root and default language:

      ASP Content Root: Directory that corresponds to the main web or domain URL

      Default Language: VB Script (Default) or JavaScript
   b. Add, delete, or move the type libraries (`dll`, `exe`, `ocx`, or `tlb`) that the ASP project depends on to compile.
7. Click **Finish**.

Adding a new C/C++ project

About this task

When you add a new C/C++ project to the application, you specify a collection of source files to scan:

- `include` path
- preprocessor definitions
- options

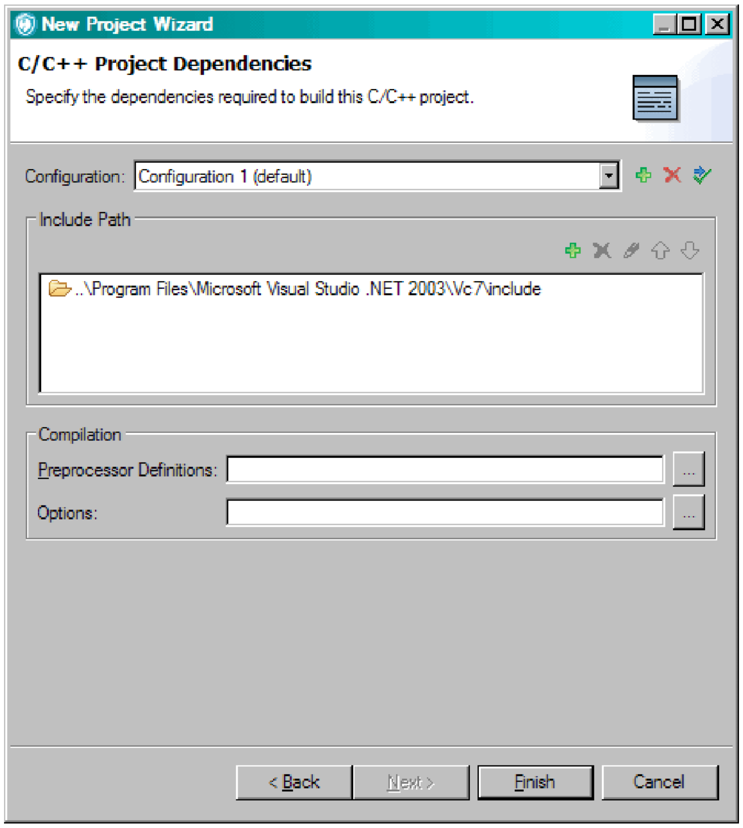
The steps in this topic direct you to complete all pages in the New Project Wizard (or New Application Wizard, if you are creating the project in it). However, some of the pages in the wizard are optional (required settings are complete when the **Finish** button is activated). Settings made in the wizard can be modified after project creation in the Properties view for a selected project. If you complete the New Project Wizard without completing optional pages, you can change the settings from those pages later on in the Properties view.

Important: In order to scan a C++ project, the project must compile and link without errors.

Procedure

1. In the Explorer view, select the application that you want to add the project to (if you have not already added an application, see "Configuring applications" on page 19).
2. Complete one of these actions to open the New Project Wizard:
   a. Select **File** > **Add Project** > **New Project** from the main workbench menu.
   b. Right-click the selected application and choose **Add Project** > **New Project** from the context menu.
3. In the Select Project Type page of the wizard, select **C/C++** as the project type and then click **Next** to advance to the next wizard page.
4. In the Project Sources wizard page:
   a. Identify the project sources. Project sources consist of the directories in which you find project files, and any additional individual files to include in the project.

      Name the project and specify the working directory. The **Working Directory** is the location in which the AppScan Source project file (`.ppf`) will reside. It is also the base for all relative paths.
   b. Click **Add Source Root** to specify a source code root and the directories or files to include or exclude from the scan. After adding the source root, you can exclude certain directories or files from it. To do this, select the directory or file (or multiselect these items) in the source root, right-click the selection, and then choose **Exclude** from the menu. If you include or exclude files, the icon to the left of the file name changes.
5. Click **Next** to advance to the next wizard page.
6. In the C/C++ Project Dependencies page, add project dependencies by specifying the project configuration and `include` path.


New Project Wizard
C/C++ Project Dependencies
Specify the dependencies required to build this C/C++ project.
Configuration:
Include Path
Compilation
Preprocessor Definitions:
Options:
< Back
Next >
Finish
Cancel

Configuration: Configuration 1 (default)

..\Program Files\Microsoft Visual Studio .NET 2003\Vc7\include

- **Configuration**: List of all available configurations for the project. Add new or delete existing configurations. Define all remaining settings for each configuration.

  You may define multiple configurations for a C/C++ project, such as `Debug` and `Release`. `Configuration 1` is the default project configuration name.
- **Include Path**: Use this section to add the fully-qualified path names to directories containing `#include` files required for the project.
- **Preprocessor Definitions**: Use this field to add preprocessing symbols defined for the project. Use these preprocessor definitions for the selected configuration.
- **Options**: Additional required compiler parameters for the project configuration.

7. Click **Finish**.

Adding a new JavaScript project

The Project Configuration Wizard helps you manually create a JavaScript project and add it to an application.

About this task

The steps in this topic direct you to complete all pages in the New Project Wizard (or New Application Wizard, if you are creating the project in it). Settings made in the wizard can be modified after project creation in the Properties view for a selected project.

Procedure

1. In the Explorer view, select the application that you want to add the project to (if you have not already added an application, see "Configuring applications" on page 19).
2. Complete one of these actions to open the New Project Wizard:
   a. Select **File** > **Add Project** > **New Project** from the main workbench menu.
   b. Right-click the selected application and choose **Add Project** > **New Project** from the context menu.
3. In the Select Project Type page of the wizard, select **JavaScript** as the project type and then click **Next** to advance to the next wizard page.
4. In the Project Sources wizard page:
   a. Identify the project sources. Project sources consist of the directories in which you find project files, and any additional individual files to include in the project.

      Name the project and specify the working directory. The **Working Directory** is the location in which the AppScan Source project file (`.ppf`) will reside. It is also the base for all relative paths.
   b. Click **Add Source Root** to specify a source code root and the directories or files to include or exclude from the scan. After adding the source root, you can exclude certain directories or files from it. To do this, select the directory or file (or multiselect these items) in the source root, right-click the selection, and then choose **Exclude** from the menu. If you include or exclude files, the icon to the left of the file name changes.
5. Click **Finish**.

Adding a new COBOL project

The Project Configuration Wizard helps you manually create a COBOL project and add it to an application.

About this task

The steps in this topic direct you to complete all pages in the New Project Wizard (or New Application Wizard, if you are creating the project in it). Settings made in the wizard can be modified after project creation in the Properties view for a selected project.

Procedure

1. In the Explorer view, select the application that you want to add the project to (if you have not already added an application, see "Configuring applications" on page 19).
2. Complete one of these actions to open the New Project Wizard:
   a. Select **File** > **Add Project** > **New Project** from the main workbench menu.
   b. Right-click the selected application and choose **Add Project** > **New Project** from the context menu.
3. In the Select Project Type page of the wizard, select **COBOL** as the project type and then click **Next** to advance to the next wizard page.
4. In the Project Sources wizard page:
   a. Identify the project sources. Project sources consist of the directories in which you find project files, and any additional individual files to include in the project.

      Name the project and specify the working directory. The **Working Directory** is the location in which the AppScan Source project file (`.ppf`) will reside. It is also the base for all relative paths.
   b. Click **Add Source Root** to specify a source code root and the directories or files to include or exclude from the scan. After adding the source root, you can exclude certain directories or files from it. To do this, select the directory or file (or multiselect these items) in the source root, right-click the selection, and then choose **Exclude** from the menu. If you include or exclude files, the icon to the left of the file name changes.
5. Click **Finish**.

Adding a new ColdFusion project

The Project Configuration Wizard helps you manually create a ColdFusion project and add it to an application.

About this task

The steps in this topic direct you to complete all pages in the New Project Wizard (or New Application Wizard, if you are creating the project in it). Settings made in the wizard can be modified after project creation in the Properties view for a selected project.

Procedure

1. In the Explorer view, select the application that you want to add the project to (if you have not already added an application, see "Configuring applications" on page 19).
2. Complete one of these actions to open the New Project Wizard:

a. Select **File** > **Add Project** > **New Project** from the main workbench menu.
b. Right-click the selected application and choose **Add Project** > **New Project** from the context menu.

3. In the Select Project Type page of the wizard, select **ColdFusion** as the project type and then click **Next** to advance to the next wizard page.
4. In the Project Sources wizard page:
   a. Identify the project sources. Project sources consist of the directories in which you find project files, and any additional individual files to include in the project.

      Name the project and specify the working directory. The **Working Directory** is the location in which the AppScan Source project file (`.ppf`) will reside. It is also the base for all relative paths.

   b. Click **Add Source Root** to specify a source code root and the directories or files to include or exclude from the scan. After adding the source root, you can exclude certain directories or files from it. To do this, select the directory or file (or multiselect these items) in the source root, right-click the selection, and then choose **Exclude** from the menu. If you include or exclude files, the icon to the left of the file name changes.
5. Click **Finish**.

Adding a new Java or JavaServer Page (JSP) project

When you add a new Java project to the application, you specify the project name, browse to the working directory, and then specify the source roots and project dependencies.

About this task

The steps in this topic direct you to complete all pages in the New Project Wizard (or New Application Wizard, if you are creating the project in it). However, some of the pages in the wizard are optional (required settings are complete when the **Finish** button is activated). Settings made in the wizard can be modified after project creation in the Properties view for a selected project. If you complete the New Project Wizard without completing optional pages, you can change the settings from those pages later on in the Properties view.

Procedure

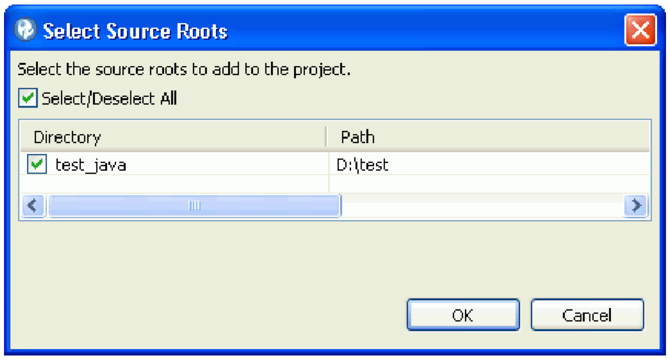
1. In the Explorer view, select the application that you want to add the project to (if you have not already added an application, see “Configuring applications” on page 19).
2. Complete one of these actions to open the New Project Wizard:
   a. Select **File** > **Add Project** > **New Project** from the main workbench menu.
   b. Right-click the selected application and choose **Add Project** > **New Project** from the context menu.
3. In the Select Project Type page of the wizard, select **Java/JSP** as the project type and then click **Next** to advance to the next wizard page.
4. In the Project Sources wizard page:
   a. Identify the project sources, which consist of the directories in which you find the project files and any additional individual files to include in the project.

      Name the project and specify the working directory. The **Working Directory** is the location of the AppScan Source project file (`.ppf`) and the base for all relative paths.

b. Add the source roots manually or allow AppScan Source for Analysis to find all valid source roots automatically.

Important:

- To analyze Java class files, they must be compiled with `javac` using the `-g` option. The AppScan Source analysis relies on the debugging information generated by this option.
- If your project contains Java source files that contain national language characters and you are running in a locale other than the native locale (for example, UTF-8), the scan will fail with errors and/or warnings in the console.
- To find the source roots automatically:
  1) Click **Find Source Roots** and browse to the root directory of the source code.
  2) From the list of all found source roots, select the source roots to add to the project.



Select the source roots to add to the project.

✓ Select/Deselect All

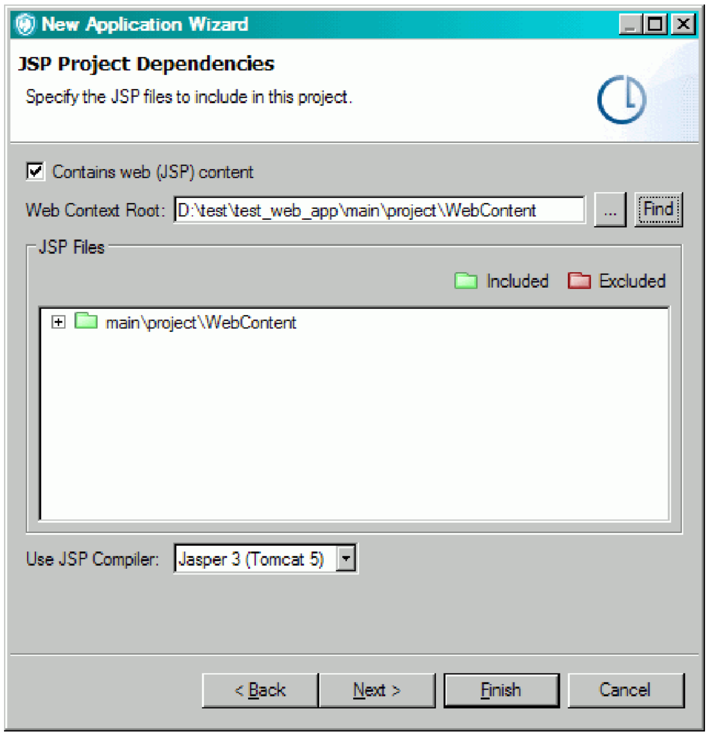
Directory	Path
✓ test_java	D:\test

  3) Click **OK**. The sources to include in the scan appear in the **Project Sources** dialog box.
- To find the source roots manually:
  1) Click **Add Source Root**.
  2) Select the source code root directory or file.
  3) Click **OK**. After adding the source root, you can exclude certain directories or files from it. To do this, select the directory or file (or multiselect these items), right-click the selection, and then choose **Exclude** from the menu. If you include or exclude files, the icon to the left of the file name changes.

Click **Finish** to add the project without setting project dependencies - or click **Next** to identify project dependencies.

5. In the JSP Project Dependencies page:

a. Identify JavaServer Page (JSP) project dependencies: For Java projects that contain JavaServer Pages, identify the JSP project dependencies. Select the **Contains web (JSP) content** check box if the project is a web application that contains JavaServer Pages.



Specify the JSP files to include in this project.

☑ Contains web (JSP) content

Web Context Root: D:\test\test_web_app\main\project\WebContent ... Find

JSP Files

Included Excluded

main\project\WebContent

Use JSP Compiler: Jasper 3 (Tomcat 5)

b. Manually select the **Web Context Root**, or click **Find** to locate it. The **Web Context Root** is a `WAR` file or a directory that contains the `WEB-INF` directory. The web context root must be the root of a valid web application.

c. Select the **JSP Compiler** for the project. Tomcat 7 (Jasper 2) is the default JSP compiler setting (the default JSP compiler can be changed in the Java and JSP preference page). To learn about the compilers that are supported by AppScan Source, see http://www.ibm.com/support/docview.wss?uid=swg27027486.

   Apache Tomcat Versions 5, 6, and 7 are included in the installation of AppScan Source. If the **Tomcat 5**, **Tomcat 6**, and **Tomcat 7** preference pages are not configured, AppScan Source will compile JSP files using the supplied Tomcat JSP compiler that is currently marked as default. If you want to employ an external Tomcat compiler, use the Tomcat preference pages to point to your local Tomcat installation.

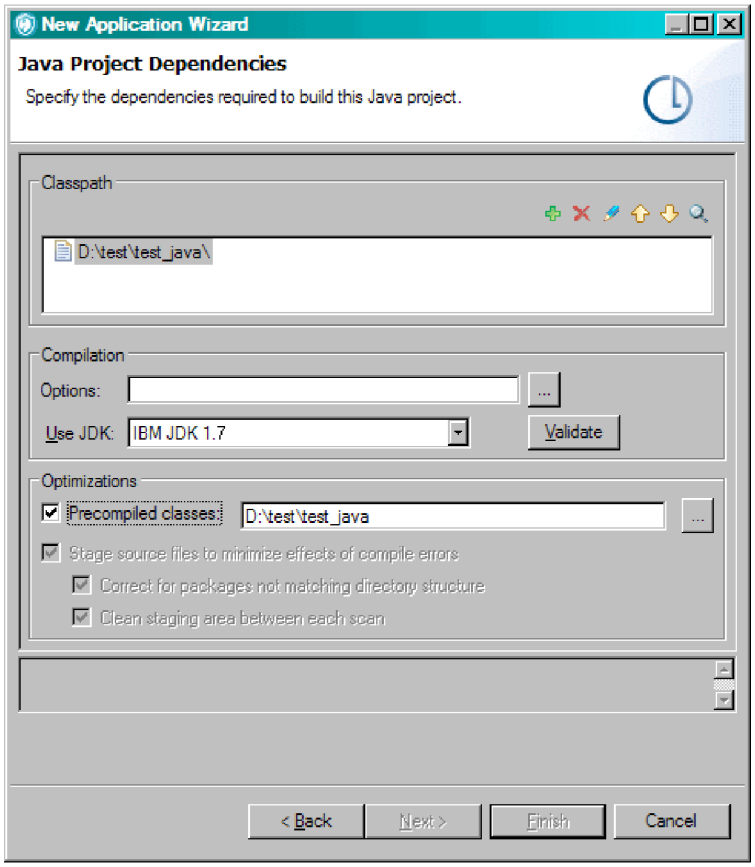
   If you are using Oracle WebLogic Server or WebSphere Application Server, you must configure the applicable preference page to point to your local installation of the application server so that it can be used for JSP compilation during analysis. If you have not already completed this configuration, you will be prompted by a message to do so when you select the JSP compiler. If you click **Yes** in the message, you will be taken to the appropriate preference page. If you click **No**, a warning link will display next to the JSP compiler selection (following the link will open the preference page).

Click **Finish** to add the project with JSP project dependencies - or click **Next** to identify Java project dependencies.

6. In the Java Project Dependencies page, identify the dependencies required to build this Java project:

   a. Add the `JAR` files manually or click **Find** for AppScan Source for Analysis to search the directories that contain the dependent `JAR` and class files.

The **Class Path** list displays the relative path to the project. The class path must specify the required JAR files and the directories containing class files that the project requires.



New Application Wizard

Java Project Dependencies

Specify the dependencies required to build this Java project.

Classpath

D:\test\test_java\

Compilation

Options:

Use JDK: IBM JDK 1.7

Validate

Optimizations

☑ Precompiled classes: D:\test\test_java

☑ Stage source files to minimize effects of compile errors

☑ Correct for packages not matching directory structure

☑ Clean staging area between each scan

< Back | Next > | Finish | Cancel

- **Add**, **Remove**, **Move Up**, and **Move Down**: Add or remove files from the class path, or move them up or down in order.
- **Find**: Find JAR and class path entries based on the source files in the project.

Important: If the Java project contains JavaServer Pages, you must also add JSP Project Dependencies.

- To find project dependencies manually:
  1) Click **Add** in the Class Path section toolbar and then select the JAR and class file directories necessary to compile the Java project.
  2) Click **OK**. The JAR files and directories appear in the class path. Change the order as necessary.
- To find dependencies automatically:
  1) Click **Find** in the Class Path section toolbar.
  2) Specify the directories in which to look for the JAR and class files necessary to compile the Java project.
  3) Select the **Look inside the source and JAR files** check box if you want AppScan Source for Analysis to find the required project dependencies based on sources and by using the provided search path.
  4) Click **Next** to find the project dependencies and identify conflicts.
- To resolve conflicts:

1) If conflicts exist, in the Resolve Conflicts dialog box, select the entry to resolve and click **Resolve** (or click **Next** to auto-resolve conflicts). A conflict occurs when AppScan Source for Analysis finds more than one `JAR` or class in a directory that satisfies the dependency.

   A red icon appears to the left of unresolved conflicts. Once resolved, the red icon changes to green and the item is **Resolved**. You may also **Remove** a conflict.

2) After you resolve or remove a conflict, you may want to verify, reorder, or remove the class path entries. Note the list of imports that could not be found. Any unresolved imports result in compilation errors when AppScan Source for Analysis scans.

b. **Options**: Specify any additional required compiler parameters for the project.

   Compilation options are the options that are passed to the compiler so that source files can compile. For example, `-source 1.5` specifies the source level of the project.

c. **Use JDK**: Specify the Java Development Kit (JDK) to use when scanning this code. The default JDK is Version 1.7. If desired, edit **Preferences** to set the default JDK and define additional JDKs.

   Note: The default compiler for JSP projects is Tomcat 7 (Jasper 2), which requires Java Version 1.6 or higher. If **Tomcat 7** is kept as default, selecting an earlier JDK (for example, **IBM JDK 1.5**) will result in compilation errors during scans.

d. The **Validate** action assures that project dependencies are correctly configured. It checks Java projects for configuration conflicts between sources and the class path, and it also checks for compilation errors. A conflict exists if a class in the class path is duplicated in the source root.

   If a conflict exists, the validation text area displays the `JAR` or location where the class is defined on the class path and whether the duplicate exists in the sources. Remove the conflict from the class path, and rerun the check.

   After checking for conflicts, **Validate** determines if the project compiles and reports any compilation errors.

e. **Precompiled classes**: This field allows you to use precompiled class files instead of compiling during a scan.

f. **Stage source files to minimize effects of compile errors**: Clear the check box if your source code compiles correctly and is arranged accurately in directories, matching the packages.

g. **Correct for packages not matching directory structure**: Select if the packages do not match the directory structure.

h. **Clean staging area between each scan**: Optimization option.

7. Click **Finish**.

Adding content to a JSP project

JavaServer Page (JSP) projects include web applications built on JavaServer Pages.

About this task

To scan JSP projects successfully, the JavaServer Pages must be in a valid web application structure. This section describes the file structure under the web context root required for a successful scan. You should be familiar with the web application structure before configuring your JSP projects.

A web application deployed into a web application server, such as Tomcat, requires a standard directory structure. The deployed application can be a set of files arranged in a directory structure or a WAR file. In the case of a WAR file, the directory structure is contained in the ZIP file, with `web context root` as the root of the directory structure.

Beneath a web context root, you find the following standard directories:

Table 6. Web context root directories

`<web-context-root>\` `WEB-INF\`	
`classes\`	Java class files arranged in directories (packages)
`lib\`	Jar files added to the class path
`web.xml`	`web.xml` describes resources available to the application

Other directories contain necessary files that may also exist. For example, you often see a directory for the content (JSP and HTML files), and for tag libraries:

Table 7. Other directories

`<web-context-root>\`	
`jsp\`	Contains JavaServer Pages in the application
`WEB-INF\`	
`tld\`	Contains tag libraries used in the application

In addition to these standard web application directories, a web application server can have special directories where it expects to find class files and JAR files shared by all deployed web applications. For example, Tomcat 7 places these JAR files in the `common\lib` or `common\endorsed` directories. The location of these nonstandard directories is specific to each application server.

Important: Before scanning JavaServer Pages, confirm that all necessary files exist in the web context root. AppScan Source for Analysis only scans JavaServer Pages in the web context root.

Procedure

1. Copy the files, if necessary, to the appropriate location under the web context root.
2. Specify the web context root as either the directory or a WAR file containing all of the JavaServer Pages.
3. Be certain that the class path includes the JAR or class file directories.
4. Configure the project properties.

Results

AppScan Source for Analysis adds the `WEB-INF\classes` directory and all JAR files in `WEB-INF\lib` to the class path, for JSP only. You can add items that are not included in the `Web-INF` path, but that are necessary to compile the JSP. These `JAR` files are similar to `weblogic.jar` or a vendor `JAR` file placed in the common directory for an application server.

JSP sources are the JavaServer Pages under the web context root that you want to scan. The source files are relative to the web context root. You are limited to the set of files within the web context root when specifying the JSP sources.

JSP project sources consist of the directories in which you find the project files and any additional individual files to include in the project.

- Specify the subset of the JavaServer Pages in the web context root. If this is not done, all files will scan.
- If the JavaServer Pages depend on Java code, you must specify these sources.
- JSP files include `jsp` and `jspx` files.

Adding a new .NET Assembly project

The New Project Wizard helps you create a .NET Assembly project. A .NET Assembly project can be used to scan compiled .NET assembly files, when source files may not be available or buildable. .NET Assembly projects consist of a working directory and list of sources, which may be directories or individual assembly files.

About this task

The steps in this topic direct you to complete all pages in the New Project Wizard (or New Application Wizard, if you are creating the project in it). Settings made in the wizard can be modified after project creation in the Properties view for a selected project.

Procedure

1. In the Explorer view, select the application that you want to add the project to (if you have not already added an application, see “Configuring applications” on page 19).
2. Complete one of these actions to open the New Project Wizard:
   a. Select **File** > **Add Project** > **New Project** from the main workbench menu.
   b. Right-click the selected application and choose **Add Project** > **New Project** from the context menu.
3. In the Select Project Type page of the wizard, select **.NET Assembly** as the project type and then click **Next** to advance to the next wizard page.
4. In the Project Sources wizard page:
   a. Identify the project sources. Project sources consist of the directories in which you find project files, and any additional individual files to include in the project.

      Name the project and specify the working directory. The **Working Directory** is the location in which the AppScan Source project file (`.ppf`) will reside. It is also the base for all relative paths.
   b. Click **Add Source Root** to specify a source code root and the directories or files to include or exclude from the scan. After adding the source root, you can exclude certain directories or files from it. To do this, select the directory or file (or multiselect these items) in the source root, right-click the selection, and then choose **Exclude** from the menu. If you include or exclude files, the icon to the left of the file name changes.

5. Click **Finish**.

Adding a new Pattern Based project

About this task

The New Project Wizard helps you manually create a Pattern Based project and add it to an application. Pattern Based projects include a collection of any language independent files for pattern-based analysis and scanning.

For example, you may want to logically group `.xml` and `.config` files and search them for certain pattern-based expressions. AppScan Source for Analysis scans the files and searches for the expressions (see "Customizing with pattern-based scan rules" on page 186 for details).

The steps in this topic direct you to complete all pages in the New Project Wizard (or New Application Wizard, if you are creating the project in it). Settings made in the wizard can be modified after project creation in the Properties view for a selected project.

Procedure

1. In the Explorer view, select the application that you want to add the project to (if you have not already added an application, see "Configuring applications" on page 19).
2. Complete one of these actions to open the New Project Wizard:
   a. Select **File** > **Add Project** > **New Project** from the main workbench menu.
   b. Right-click the selected application and choose **Add Project** > **New Project** from the context menu.
3. In the Select Project Type page of the wizard, select **Pattern Based** as the project type and then click **Next** to advance to the next wizard page.
4. In the Project Sources wizard page:
   a. Identify the project sources. Project sources consist of the directories in which you find project files, and any additional individual files to include in the project.

      Name the project and specify the working directory. The **Working Directory** is the location in which the AppScan Source project file (`.ppf`) will reside. It is also the base for all relative paths.
   b. Click **Add Source Root** to specify a source code root and the directories or files to include or exclude from the scan. After adding the source root, you can exclude certain directories or files from it. To do this, select the directory or file (or multiselect these items) in the source root, right-click the selection, and then choose **Exclude** from the menu. If you include or exclude files, the icon to the left of the file name changes.
5. Click **Finish**.

Adding a new Perl project

The New Project Wizard helps you manually create a Perl project and add it to an application.

Procedure

1. In the Explorer view, select the application that you want to add the project to (if you have not already added an application, see "Configuring applications" on page 19).

2. Complete one of these actions to open the New Project Wizard:
   a. Select **File** > **Add Project** > **New Project** from the main workbench menu.
   b. Right-click the selected application and choose **Add Project** > **New Project** from the context menu.
3. In the Select Project Type page of the wizard, select **Perl** as the project type and then click **Next** to advance to the next wizard page.
4. In the Project Sources wizard page:
   a. Identify the project sources. Project sources consist of the directories in which you find project files, and any additional individual files to include in the project.

      Name the project and specify the working directory. The **Working Directory** is the location in which the AppScan Source project file (`.ppf`) will reside. It is also the base for all relative paths.
   b. Click **Add Source Root** to specify a source code root and the directories or files to include or exclude from the scan. After adding the source root, you can exclude certain directories or files from it. To do this, select the directory or file (or multiselect these items) in the source root, right-click the selection, and then choose **Exclude** from the menu. If you include or exclude files, the icon to the left of the file name changes.
5. Click **Finish**.

PHP project configuration

When you add a new PHP: Hypertext Preprocessor (PHP) project to the application, you specify the project name, browse to the working directory, and then specify the source roots and project dependencies. The project dependencies can also be set in the Project Dependencies tab of the project properties after the project has been created.

About this task

The steps in this topic direct you to complete all pages in the New Project Wizard (or New Application Wizard, if you are creating the project in it). However, some of the pages in the wizard are optional (required settings are complete when the **Finish** button is activated). Settings made in the wizard can be modified after project creation in the Properties view for a selected project. If you complete the New Project Wizard without completing optional pages, you can change the settings from those pages later on in the Properties view.

Note: For PHP, VB6, and Classic ASP, only ISO-8859-1 (Western Europe), UTF-8, and UTF-16 character sets are supported.

Procedure

1. In the Explorer view, select the application that you want to add the project to (if you have not already added an application, see “Configuring applications” on page 19).
2. Complete one of these actions to open the New Project Wizard:
   a. Select **File** > **Add Project** > **New Project** from the main workbench menu.
   b. Right-click the selected application and choose **Add Project** > **New Project** from the context menu.
3. In the Select Project Type page of the wizard, select **PHP** as the project type and then click **Next** to advance to the next wizard page.
4. In the Project Sources wizard page:

   a. Identify the project sources. Project sources consist of the directories in which you find project files, and any additional individual files to include in the project.

      Name the project and specify the working directory. The **Working Directory** is the location in which the AppScan Source project file (`.ppf`) will reside. It is also the base for all relative paths.
   b. Click **Add Source Root** to specify a source code root and the directories or files to include or exclude from the scan. After adding the source root, you can exclude certain directories or files from it. To do this, select the directory or file (or multiselect these items) in the source root, right-click the selection, and then choose **Exclude** from the menu. If you include or exclude files, the icon to the left of the file name changes.
5. **PHP Project Configuration**: In the **PHP Document Root** field, enter or browse to a directory that represents the root of your PHP application. This is the file system directory that is mapped to the site's base URL. If a PHP document root is not specified, the source root that was specified in the Project Sources page will be used.
6. Optional: Set an **Include Path**. Include path directories are used to resolve relative paths to files used in PHP `include` statements (for example, `include`, `include_once`, `require`, `require_once`).
7. Optional: Set a **Class Include Path**. Class include path directories are used to find files that contain PHP class definitions.
8. Click **Finish**.
9. Optional: **Configure unresolved dependencies**: In the project properties, go to the Project Dependencies page and follow the steps for "Configuring unresolved PHP `include` expressions" on page 45 and "Configuring unresolved PHP class references" on page 49.

Example: Creating a new PHP project

About this task

This example shows you how to use the New Application Wizard to create a PHP project.

Procedure

1. Complete one of these actions:
   - Select **File** > **Add Application** > **Create a new application** from the main menu bar.
   - In the Explorer view toolbar, click the **Add Application Menu** down-arrow button and select **Create a new application** from the menu.
   - In the Explorer view, right-click **All Applications** and then select **Add Application** > **Create a new application** from the menu.
2. Enter a **Name** for the application.
3. Browse to the **Working Directory** in which to save the application. The new application file name extension will be `.paf`.
4. Click **Next** to configure the project.
5. In the Select Project Type page of the wizard, select **PHP** as the project type and then click **Next** to advance to the next wizard page.
6. In the Project Sources page:
   a. In the **Name** field, enter a name for the project - for example `MyProject`.
   b. In the **Working Directory** field, browse to the location in which you want to store the project file that will be created - for example, `C:\Apps\MyProject`.

   c. Click **Add Source Root** to add all directories that contain PHP files that should be scanned. For example, in the Select a File or Directory dialog box, browse to `C:\Apps\MyProject\root` and then click **OK** to close the dialog box.

   Click **Next**.
7. In the PHP Project Configuration page:
   a. In the **PHP Document Root** field, enter or browse for the directory that represents the root of your PHP application. This is the file system directory that is mapped to the site's base URL. By default, this field pre-populates with the source root that was specified in the Project Sources page.
   b. Optional: Add include path directories. These are used to resolve relative paths to files used in PHP Include statements (for example, `include`, `include_once`, `require`, `require_once`).
   c. Optional: Add class path directories. These are used to find files that contain PHP class definitions.
8. Click **Finish**. You now have a PHP project that is ready to be scanned.

Configuring unresolved PHP `include` expressions

Before you begin

In the project properties, go to the Project Dependencies page.

Procedure

1. Click **Configure Unresolved Include Expressions** to open the Configure Unresolved Include Expressions dialog box.
2. The upper portion of the dialog box lists the unresolved `include` expressions (all `include` expressions that were not resolved or that required extra processing to resolve). Information provided about the expressions includes:

Option	Description
Included Text / Updated Text	This column displays the expression text as it exists in the source code. This column can be expanded by clicking **+**, which will then show the updated text that was used during the last scan. It will display `<empty>` if there was no updated text available during the last scan. There may be multiple updated text lines displayed when expanded - one for each place in the source code where this expression was used.
Status	This column will contain an `X` for expressions that are unresolved or a check mark for expressions that were successfully resolved.

Option	Description
Resolved By	This column indicates how the updated text was generated. Values include: • `AutoResolver`: The application used internal heuristics to find the file. • `SearchReplace`: One or more search and replace rules were applied to the included text to generate the updated text. • `SearchReplace+AutoResolver`: One or more search and replace rules were applied to the included text to generate the updated text and then internal heuristics were applied to find the file. • `SearchReplace+IncludePath`: One or more search and replace rules were applied to the included text to generate the updated text and then was combined with a directory on the `include` path to find the file. • `SearchReplace+RelativeDir`: One or more search and replace rules were applied to the included text to generate the updated text and then was found relative to the source directory of the file containing the `include` expression.
Source File, Line, Column	These columns show the location in the source code in which the expression is used. You may want to look at these locations in an editor to see how they should be resolved.

Note: Some of the columns may be blank. This is because included text can have multiple updated text lines when expanded. These columns will have appropriate text for each of these updated text lines.

3. In the lower portion of the dialog box, the Include Path tab contains the same `include` path information that was entered in the PHP Project Dependencies page. You can update this information in this dialog box (while viewing the unresolved `include` expressions).
4. In the lower portion of the dialog box, the Search and Replace tab is used for adding rules for replacing dynamic text in an `include` expression with static text that is a full or partial file path to an `include` file. There are three columns in the Search and Replace table:

Option	Description
Command	The value in this column determines how the **Search Text** and **Replacement Text** columns are used. The choices are: • **Replace Text**: This command is used for simple text search and replace. The **Search Text** is used as is and, if it is found anywhere in the included text, it is replaced with the **Replacement Text**. • **Replace Function**: This command is used when a function call is to be replaced. The **Search Text** should be the name of the function without parentheses. The search text will be augmented to look for a function call with the specified name followed by parentheses and will match anything within the parentheses. • **Replace RegEx**: This is an advanced feature that allows a regular expression to be specified for the search text.
Search Text	This is the text to search for in the `include` expression. You can select text within an `include` expression and copy it to the clipboard and then paste it here. See the above description of the **Command** column for the variations in specifying the search text.
Replacement Text	This is the text used to replace the search text. This is static text that is a full or partial file path to an `include` file. There are also a few variables that can be placed in the replacement text. They can be typed directly into the **Replacement Text** cell or selected from the **Replacement Text Variables** menu above the table (this will copy the selected variable to the clipboard). The variables that can be selected from the **Replacement Text Variables** menu list are: • `%ROOT_DIR%`: This variable will be replaced by the PHP document root directory specified for the project. • `%SRC_DIR%`: This variable will be replaced by the directory of the file that contains the `include` expression. • `%ARG_N%`: This variable only applies when the command is **Replace Function**. The `N` in the variable should be replaced with an integer (for example, `%ARG_1%` or `%ARG_2%`). This variable will then be replaced by the text that is passed into the function call's `N`th parameter.

The rules are applied in order. After each successful search and replace operation, the new text is checked to see if the file can be found. If the file is not found then the next rule is tried against the updated text.

Every PHP project starts with three search and replace rules that try to replace some standard PHP constants and functions that are commonly used in `include` expressions.

Example: Configuring unresolved `include` expressions:
Before you begin

Unresolved `include` expressions are configured in the Project Dependencies tabbed page of the project properties. Once in the page, click **Configure Unresolved Include Expressions** to open the Configure Unresolved Include Expressions dialog box.

This example assumes that you have added an `include` path to your project (this can be done while creating the project or while in the Project Dependencies page) and that you have run a scan. After the scan has completed, open the Configure Unresolved Include Expressions dialog box to see the **Unresolved Include Expressions** list. In this example, `MYPROJECT_ROOT_PATH.'/a/b/filename.php'`, `MYPROJECT_ROOT_PATH.'/language/'.$configInfo['language'].'/mypage.php'`, and `configGet('database_inc','./includes/database.inc')` are expressions in that list.

Procedure

1. Replace a leading PHP constant or variable with a directory by following these steps:
   a. Select `MYPROJECT_ROOT_PATH.'/a/b/filename.php'` - this causes the text in the expression to be selected. You can then use the mouse or cursor keys to select a portion of the expression. Select `MYPROJECT_ROOT_PATH` and then right-click and choose **Copy**.
   b. Select the Search and Replace tab.
   c. Click the **Add Rule for Selected Unresolved Item** button (decorated with a green **plus** symbol). This adds a new search and replace rule to the list.
   d. In the new rule, select `NewSearchText` and then right-click and choose **Paste** from the menu. This causes `NewSearchText` to be replaced with `MYPROJECT_ROOT_PATH`.
   e. From the **Replacement Text Variables** menu, select `%ROOT_DIR%`. This causes the `%ROOT_DIR%` text string to be copied to the clipboard.
   f. In the rule, select `NewReplacementText` and then right-click and choose **Paste** from the menu. This causes `NewReplacementText` to be replaced with `%ROOT_DIR%`.

   You now have a new rule that will replace a constant with the path to the PHP document root directory. The PHP concatenation operator (`.`) and the text string that follows it will be combined with the replacement text to generate a single path expression. The next time the project is scanned, the `include` expressions that use this constant should succeed.
2. To replace a dynamic expression with a single value:
   a. Select `MYPROJECT_ROOT_PATH.'/language/'.$configInfo['language'].'/mypage.php'` - this causes the text in the expression to be selected. You can then use the mouse or cursor keys to select a portion of the expression. Select `$configInfo['language']` and then right-click and choose **Copy**.
   b. Select the Search and Replace tab.
   c. Click the **Add Rule for Selected Unresolved Item** button (decorated with a green **plus** symbol). This adds a new search and replace rule to the list.

d. In the new rule, select `NewSearchText` and then right-click and choose **Paste** from the menu. This causes `NewSearchText` to be replaced with `$configInfo['language']`.
e. In the rule, select `NewReplacementText` and replace it with new text by typing `english`.

You now have a new rule that will replace the expression with the specified value. The PHP concatenation operator (`.`) will be applied to generate a single path expression. The next time the project is scanned, the `include` expressions that use this expression should succeed.

3. To replace a PHP function call with one of its arguments:
   a. Select `configGet('database_inc','./includes/database.inc')` - this causes the text in the expression to be selected. You can then use the mouse or cursor keys to select a portion of the expression. Select `configGet` and then right-click and choose **Copy**.
   b. Select the Search and Replace tab.
   c. In the new rule, select `Replace Text` in the first column and then select `Replace Function` from the menu.
   d. In the rule, select `NewSearchText` and then right-click and choose **Paste** from the menu. This causes `NewSearchText` to be replaced with `configGet`.
   e. From the **Replacement Text Variables** menu, select `%ARG_1%`. This causes the variable to be copied to the clipboard.
   f. In the rule, select `NewReplacementText` and then right-click and choose **Paste** from the menu. Edit the pasted text to be `%ARG_2%` instead of `%ARG_1%`.

   You now have a new rule that will replace a function call with the value of its second parameter. The next time the project is scanned, the `include` expressions that use this function call should succeed.

Configuring unresolved PHP class references

Before you begin

In the project properties, go to the Project Dependencies page.

Procedure

1. Click **Configure Unresolved Class References** to open the Configure Unresolved Class References dialog box.
2. The upper portion of the dialog box lists the unresolved class references (all class references that were not resolved during the last scan). Information provided about the class references includes:

Option	Description
Class Name / Generated File Name	This column displays the class name that was referenced in the source code. This column can be expanded by clicking **+**, which will then show the generated file name that was used during the last scan to try and find the class. There may be multiple file names when it is expanded - one for each place in the source code where this class was used.
Status	This column will contain an `X` for classes that are unresolved or a check mark for classes that were successfully resolved.

Option	Description
Resolved By	This column indicates how the generated filename was created. Values include: • `AutoResolver`: The application used internal heuristics to find the file. • `SearchReplace`: One or more search and replace rules were applied to the class name to create the generated file name.
Source File, Line, Column	These columns show the location in the source code in which the class is used. You may want to look at these locations in an editor to see how they should be resolved.

Note: Some of the columns may be blank. This is because class names can have multiple generated file name lines when expanded. These columns will have appropriate text for each of these generated file name lines.

3. In the lower portion of the dialog box, the Class Include Path tab contains the same class `include` path information that was entered in the PHP Project Dependencies page. You can update this information in this dialog box (while viewing the unresolved class references).
4. In the lower portion of the dialog box, the Search and Replace tab is used for adding rules for modifying an unresolved class name to a full or partial file path containing the definition of that class. There are three columns in the Search and Replace table:

Option	Description
Command	The value in this column determines how the **Search Text** and **Replacement Text** columns are used. The choices are: • **Match Text**: This command is used for text search and replace. The `*` character is allowed in the search text and will match 0 or more characters. The results of a match text will not affect any other search and replace rules. This is usually used to try adding an extension to the class name or to filter out prefixes and suffixes from the class name when generating a file name. • **Replace Text**: This command is used for simple text search and replace. The **Search Text** is used as is and if it is found anywhere in the class name, it is replaced with the **Replacement Text**. This is used to modify the class name for use in the following rules. • **Replace RegEx**: This is an advanced feature that allows a regular expression to be specified for the search text.
Search Text	This is the text to search for in the class reference. You can select text within a class name and copy it to the clipboard and then paste it here. See the above description of the **Command** column for the variations in specifying the search text.

Option	Description
Replacement Text	This is the text used to replace the search text. This is static text that is a full or partial file path to a file containing the definition of the class. There are also a few variables that can be placed in the replacement text. They can be typed directly into the **Replacement Text** cell or selected from the **Replacement Text Variables** menu above the table (this will copy the selected variable to the clipboard). The variables that can be selected from the **Replacement Text Variables** menu list are: • `%ROOT_DIR%`: This variable will be replaced by the PHP document root directory specified for the project. • `%SRC_DIR%`: This variable will be replaced by the directory of the file that contains the reference to the class. • `%MATCH_N%`: This variable only applies when the command is **Match Text**. The `N` in the variable should be replaced with an integer (for example, `%MATCH_1%` or `%MATCH_2%`). This variable will then be replaced by the text that matches the `Nth` `*` in the search text.

The rules are applied in order. After each successful search and replace operation, the new text is checked to see if the file can be found. If the file is not found then the next rule is tried against the updated text, except if the command is **Match Text**.

Every PHP Project starts with two simple Search/Replace rules that try to add some common file extensions onto the class name.

5. In the lower portion of the dialog box, the Found Classes tab lists all classes that were found during the last scan. This can be used to update the class `include` path and search and replace rules. You can select an unresolved class reference in the Unresolved Class References section of the dialog and click **Find Declaration**. If the declaration is found, it will display in the Found Classes tab list.

Example: Configuring unresolved class references:
Before you begin

Unresolved class references are configured in the Project Dependencies tabbed page of the project properties. Once in the page, click **Configure Unresolved Class References** to open the Configure Unresolved Class References dialog box.

This example assumes that you have added a class `include` path to your project (this can be done while creating the project or while in the Project Dependencies page) and that you have run a scan. After the scan has completed, open the Configure Unresolved Class References dialog box to see the **Unresolved Class References** list.

In these examples, you are providing replacement text values. Note that these can have multiple variables in the text - for example, `%ROOT_DIR%/modules/%MATCH_1%/classes/%MATCH_1%.class.inc`.

Procedure

1. To add another file extension that is used by `include` files:
   a. Select the Search and Replace tab.
   b. Click the **Add Rule for Selected Unresolved Item** button (decorated with a green **plus** symbol). This adds a new search and replace rule to the list.
   c. In the new rule, select the **Replacement Text**, `%MATCH_1%.php`. In this string, delete `.php` and type `.class.inc` in its place. The **Replacement Text** should now be `%MATCH_1%.class.inc`.

   You now have a new rule that will try to add the suffix `.class.inc` to a class name when resolving the class.
2. To remove a prefix from a class name:
   a. Select the Search and Replace tab.
   b. Click the **Add Rule for Selected Unresolved Item** button (decorated with a green **plus** symbol). This adds a new search and replace rule to the list.
   c. In the new rule, select the string in the **Search Text** column (`*`) and type `Abc*` in its place.
   d. Do not alter the `%MATCH_1%.php` replacement text.

   You now have a new rule that will map class names like `AbcHello` to `Hello.php`.
3. To remove a suffix from a class name:
   a. Select the Search and Replace tab.
   b. Click the **Add Rule for Selected Unresolved Item** button (decorated with a green **plus** symbol). This adds a new search and replace rule to the list.
   c. In the new rule, select the string in the **Search Text** column (`*`) and type `*Xyz` in its place.
   d. Do not alter the `%MATCH_1%.php` replacement text.

   You now have a new rule that will map class names like `ByeByeXyz` to `ByeBye.php`.
4. You may want to map class names like `Abc_Def_Ghi_class` so that their prefixes are used as relative paths into the file system (for example `Abc/Def/Ghi/class.php`). To modify class name text for use in other rules:
   a. Select the Search and Replace tab.
   b. Click the **Add Rule for Selected Unresolved Item** button (decorated with a green **plus** symbol). This adds a new search and replace rule to the list.
   c. In the new rule, select **Match Text** in the first column and then select **Replace Text** from the menu.
   d. In the rule, select the string in the **Search Text** column (`*`) and type **_** in its place.
   e. Select the string in the **Replacement Text** column and type `/` in its place.
   f. While the rule is selected, click **Move Up** to move this rule to the top of the list.

   You now have a new rule that will replace underscores (_) with slashes (/) and the updated text will be used by all subsequent rules. This rule will change `Abc_Def_Ghi_class` to `Abc/Def/Ghi/class` and then the remaining **Match Text** rules will try to add extensions such as `.php` and `.inc`.

Adding a new PL/SQL project

The New Project Wizard helps you manually create a PL/SQL project and add it to an application.

About this task

The steps in this topic direct you to complete all pages in the New Project Wizard (or New Application Wizard, if you are creating the project in it). Settings made in the wizard can be modified after project creation in the Properties view for a selected project.

Procedure

1. In the Explorer view, select the application that you want to add the project to (if you have not already added an application, see "Configuring applications" on page 19).
2. Complete one of these actions to open the New Project Wizard:
   a. Select **File** > **Add Project** > **New Project** from the main workbench menu.
   b. Right-click the selected application and choose **Add Project** > **New Project** from the context menu.
3. In the Select Project Type page of the wizard, select **PL/SQL** as the project type and then click **Next** to advance to the next wizard page.
4. In the Project Sources wizard page:
   a. Identify the project sources. Project sources consist of the directories in which you find project files, and any additional individual files to include in the project.

      Name the project and specify the working directory. The **Working Directory** is the location in which the AppScan Source project file (`.ppf`) will reside. It is also the base for all relative paths.
   b. Click **Add Source Root** to specify a source code root and the directories or files to include or exclude from the scan. After adding the source root, you can exclude certain directories or files from it. To do this, select the directory or file (or multiselect these items) in the source root, right-click the selection, and then choose **Exclude** from the menu. If you include or exclude files, the icon to the left of the file name changes.
5. Click **Finish**.

Adding a new T-SQL project

The New Project Wizard helps you manually create a T-SQL project and add it to an application.

About this task

The steps in this topic direct you to complete all pages in the New Project Wizard (or New Application Wizard, if you are creating the project in it). Settings made in the wizard can be modified after project creation in the Properties view for a selected project.

Procedure

1. In the Explorer view, select the application that you want to add the project to (if you have not already added an application, see "Configuring applications" on page 19).
2. Complete one of these actions to open the New Project Wizard:

   a. Select **File** > **Add Project** > **New Project** from the main workbench menu.
   b. Right-click the selected application and choose **Add Project** > **New Project** from the context menu.
3. In the Select Project Type page of the wizard, select **T-SQL** as the project type and then click **Next** to advance to the next wizard page.
4. In the Project Sources wizard page:
   a. Identify the project sources. Project sources consist of the directories in which you find project files, and any additional individual files to include in the project.

      Name the project and specify the working directory. The **Working Directory** is the location in which the AppScan Source project file (`.ppf`) will reside. It is also the base for all relative paths.
   b. Click **Add Source Root** to specify a source code root and the directories or files to include or exclude from the scan. After adding the source root, you can exclude certain directories or files from it. To do this, select the directory or file (or multiselect these items) in the source root, right-click the selection, and then choose **Exclude** from the menu. If you include or exclude files, the icon to the left of the file name changes.
5. Click **Finish**.

Adding a new Visual Basic project

About this task

The steps in this topic direct you to complete all pages in the New Project Wizard (or New Application Wizard, if you are creating the project in it). Settings made in the wizard can be modified after project creation in the Properties view for a selected project.

Note: For PHP, VB6, and Classic ASP, only ISO-8859-1 (Western Europe), UTF-8, and UTF-16 character sets are supported.

Procedure

1. In the Explorer view, select the application that you want to add the project to (if you have not already added an application, see “Configuring applications” on page 19).
2. Complete one of these actions to open the New Project Wizard:
   a. Select **File** > **Add Project** > **New Project** from the main workbench menu.
   b. Right-click the selected application and choose **Add Project** > **New Project** from the context menu.
3. In the Select Project Type page of the wizard, select **Visual Basic** as the project type and then click **Next** to advance to the next wizard page.
4. In the Project Sources wizard page:
   a. Identify the project sources. Project sources consist of the directories in which you find project files, and any additional individual files to include in the project.

      Name the project and specify the working directory. The **Working Directory** is the location in which the AppScan Source project file (`.ppf`) will reside. It is also the base for all relative paths.
   b. Click **Add Source Root** to specify a source code root and the directories or files to include or exclude from the scan. After adding the source root, you can exclude certain directories or files from it. To do this, select the

directory or file (or multiselect these items) in the source root, right-click the selection, and then choose **Exclude** from the menu. If you include or exclude files, the icon to the left of the file name changes.

5. Click **Finish**.

Copying projects

AppScan Source for Analysis allows you to copy all project types except .NET projects. Modifications to the project do not affect the duplicated project; after you copy a project, there is no connection between the original project and the copied project. When you copy an imported project, you create an AppScan Source project file (`.ppf`) with all configuration information.

Procedure

1. In the Explorer view, right-click the project that you want to copy and then select **Copy Project** in the menu.
2. In the Copy Project dialog box:
   a. Name the new project.
   b. Identify the destination application for the duplicated project (the destination application must be a manually-created AppScan Source application or one that was created using the Application Discovery Assistant).
   c. Identify a destination directory (working directory for the new project).

Modifying application and project properties

When you select an application or project in the Explorer view, the current properties appear in the Properties view, where you can make modifications.

About this task

"Properties view: selected application" on page 193 and "Properties view: selected project" on page 195 provide detailed information about the settings that can be modified in the Properties view when an application or project is selected.

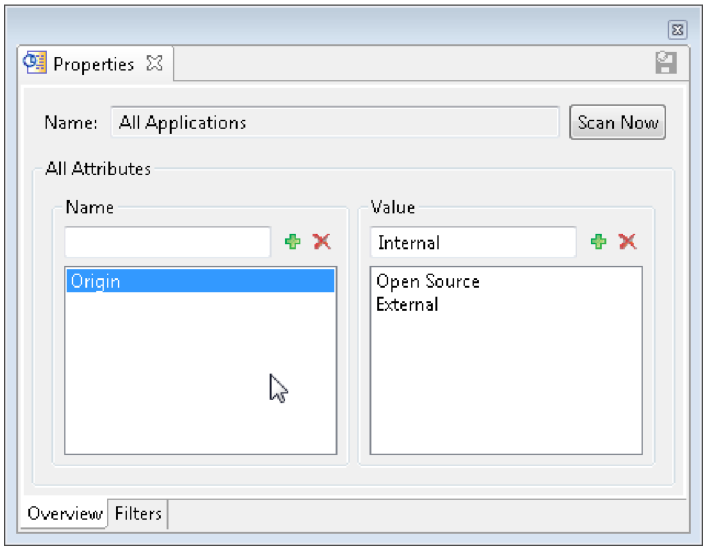
Procedure

1. Select the application or project in the Explorer view.
2. Review the properties in the Properties view.
3. Make the changes on the appropriate tab pages. The available properties pages are language-dependent.
4. Click **Save**.

Global attributes

Global attributes must be defined before they can be associated with individual applications. Global attributes are defined in the Properties view by selecting **All Applications** in the Explorer view.

About this task



To delete an attribute or its value, select the name or value and click **Delete Attribute** (✕). Deleting an attribute does not affect historical results.

To create an attribute and make it available to any application:

Procedure

1. Select **All Applications** in the Explorer view.
2. Open the **Overview** tab in the Properties view.
3. Type a name for the attribute and click **Add Attribute** (✚) - or click **Add Attribute** without specifying a name first (you will then be prompted by dialog box to enter a name for the attribute).
4. Type a **Value** for the attribute and click **Add Attribute Value** - or click **Add Attribute Value** without specifying a value first (you will then be prompted by dialog box to add a value).
5. Repeat these steps to add multiple attribute values.

Application attributes

Application attributes apply to the currently-selected application and depend on previously created global attributes.

Procedure

1. Select the application in the Explorer view.
2. Open the **Overview** tab in the Properties view.
3. Click **Add Attributes**. The **Global Attributes** dialog box appears with a list of previously-created attributes (instructions for creating global attributes can be found in “Global attributes” on page 55).
4. Double-click the attribute that you want to add - or select it and click **OK**. The attribute is added to the Application Attributes section of the Properties view.
5. Click the **Value** column and select a value for this application from the list (multiple values are available if the global attribute was created with multiple values). You can associate multiple attributes to an application.

Removing applications and projects

You can remove applications and projects from AppScan Source for Analysis if they are not registered.

Procedure

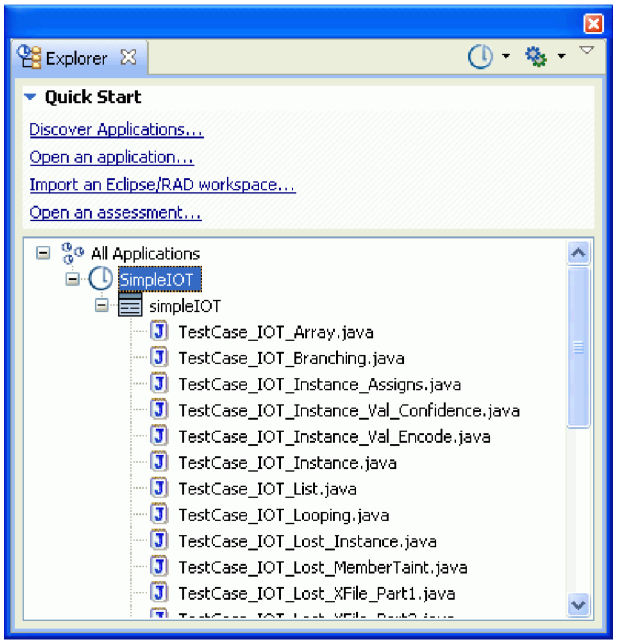
1. Select the application or project that you want to remove. Multiple applications and multiple projects can be selected for removal, however, a mix of applications and projects cannot be selected for removal.
2. Complete one of these actions:
   - Right-click the selection and choose **Remove Application** or **Remove Project** from the menu.
   - Press the keyboard Delete key.
   - Select **Edit** > **Remove** from the main workbench menu.

Explorer view

The Explorer view contains a **Quick Start** section at the top - and an explorer section at the bottom which contains one node, **All Applications**. The **Quick Start** section contains several useful links that launch common actions. The explorer section consists of a tree pane that provides a hierarchical view of your resources: applications, projects, directories, and project files, with **All Applications** as its root. You navigate these resources much like a file browser. As you navigate the view, the selection state of the tree determines the available tabs in the Properties view.

- “General information” on page 58
- “Quick Start section” on page 58
- “Toolbar buttons” on page 59
- “Right-click menu options” on page 59
- “Application and project indicators” on page 62

General information


Explorer
Quick Start
Discover Applications...
Open an application...
Import an Eclipse/RAD workspace...
Open an assessment...
All Applications
SimpleIOT
simpleIOT
TestCase_IOT_Array.java
TestCase_IOT_Branching.java
TestCase_IOT_Instance_Assigns.java
TestCase_IOT_Instance_Val_Confidence.java
TestCase_IOT_Instance_Val_Encode.java
TestCase_IOT_Instance.java
TestCase_IOT_List.java
TestCase_IOT_Looping.java
TestCase_IOT_Lost_Instance.java
TestCase_IOT_Lost_MemberTaint.java
TestCase_IOT_Lost_XFile_Part1.java

In the Explorer view, you add applications and projects and scan code using toolbar buttons, links in the **Quick Start** section, or right-click menu commands in the explorer section. Once you have added applications, the explorer section provides visual indicators of your applications and projects and the status of each.

Tip: In the Explorer view, hover help is available to indicate the file name and path of applications, projects, and files. Hover help also indicates if an application or project is registered.

Quick Start section

The **Quick Start** section offers these links for launching common tasks:

- **Discover applications**: This launches the Application Discovery Assistant, which allows to you to quickly create and configure applications and projects for Java and Microsoft Visual Studio source code.
- **Open an application**: This launches an Open dialog box, which allows you to browse for and add an existing application to the set of applications. File types that can be added include `.paf`, `.sln`, `.dsw`, and `.ewf`.
- **Import an Eclipse/RAD workspace**: This launches the Add Workspace dialog box, which allows you to add an existing Eclipse or IBM Rational Application Developer for WebSphere Software (RAD) workspace that contains Java projects. After the workspace has been imported, you will be able to scan any Java projects that it contains.

  Note: Before importing a workspace, be certain that you have installed and updated the development environment as described in "Configuring your development environment for Eclipse and Rational Application Developer for WebSphere Software (RAD) projects" on page 26.
- **Open an assessment**: This launches an Open dialog box, which allows you to browse for an AppScan Source assessment file. File types that can be opened include `.ozasmt` and `.xml`.

Toolbar buttons

Table 8. Toolbar buttons

Action	Icon	Description
Add Application Menu	🕒	Clicking the down-arrow on the **Add Application Menu** button allows you to select actions for creating a new application, opening an existing application, importing a workspace, or launching the Application Discovery Assistant.
Scan Selection	⚙	The **Scan Selection** button allows you to scan the object that is selected in the explorer section. The default scan configuration will be used for the scan. To choose a different scan configuration to use for the scan, click the down-arrow on the **Scan Selection** button. Select the scan configuration that you want to use - or choose the **Edit Configurations** action to set a different scan configuration as default (in the Scan Configuration view, select the configuration that you want to set as default, and then click **Select as Default**).
View Menu		The **View Menu** button opens a menu that allows you to refresh the explorer section and hide registered items.

Right-click menu options

The availability of right-click menu options is determined by the item that is selected in the explorer section.

- When **All Applications** is selected in the explorer section, these right-click menu options are available:
  - **Scan All Applications**: Scan all applications. The scan will run with the default scan configuration.
  - **Scan All Applications With**: Select the scan configuration that you want to use - or choose the **Edit Configurations** action to set a different scan configuration as default (in the Scan Configuration view, select the configuration that you want to set as default, and then click **Select as Default**).
  - **Add Application**
    - **Create a new application**: Add a new application to the set of applications. This action launches the New Application Wizard.

- **Open an existing application**: This launches an Open dialog box, which allows you to browse for and add an existing application to the set of applications. File types that can be added include `.paf`, `.sln`, `.dsw`, and `.ewf`.
- **Import an existing Eclipse/ RAD workspace**: This launches the Add Workspace dialog box, which allows you to add an existing Eclipse or IBM Rational Application Developer for WebSphere Software (RAD) workspace that contains Java projects. After the workspace has been imported, you will be able to scan any Java projects that it contains.

  Note: Before importing a workspace, be certain that you have installed and updated the development environment as described in "Configuring your development environment for Eclipse and Rational Application Developer for WebSphere Software (RAD) projects" on page 26.
- **Discover applications**: This launches the Application Discovery Assistant, which allows to you to quickly create and configure applications and projects for Java and Microsoft Visual Studio source code.

– **Expand All**

– **Collapse All**

– **Properties**: Selecting this opens the Properties view for the selected item.

• When an application is selected in the explorer section, these right-click menu options are available:

– **Scan Application**: Scan the selected application, project, or file. The scan will run with the default scan configuration.

– **Scan Application With**: Select the scan configuration that you want to use - or choose the **Edit Configurations** action to set a different scan configuration as default (in the Scan Configuration view, select the configuration that you want to set as default, and then click **Select as Default**).

– **Add Project**

- **New Project**: If an application is selected in the Explorer view, this action is available and choosing it allows you to add a new project to the application. This action launches the New Project Wizard.
- **Existing Project**: If an application is selected in the Explorer view, this action is available and choosing it allows you to add an existing project to the application. This action launches a dialog box that allows you to browse for a `.ppf` , `.vcproj`, `.vcxproj`, `.csproj`, `.vbproj`, `.dsp`, or `.epf` file to open.
- **Multiple Projects**: Add multiple projects to the application that is selected in the Explorer view. This action launches a dialog box that allows you to complete one of these tasks:
  - Specify a directory in which to search for projects.
  - Specify a workspace in which to search for projects.
  - Specify a Microsoft Solution file in which to search for projects.

  In the search results, you can select one or more projects to add.

– **Remove Application**: If an application is selected in the Explorer view, this action is available and choosing it removes the selected application.

– **Add Custom Finding**: This action launches the Create Custom Finding dialog box, allowing you to create a custom finding for the selected application.

– **Refresh**: Refresh the contents of a selected application, project, or view.

– Register/unregister:

    - **Register Application**: Register the selected application or project with AppScan Source. You must register applications and projects before they can be published to the AppScan Source Database.
    - **Register Application As...**: Select this to reregister an application with a new name.
    - **Unregister Application**: Unregister the selected application or project.
    - **Locate**: Select this to associate a local application or project with one that has been registered by another AppScan Source user.
  - **Expand All**
  - **Collapse All**
  - **Properties**: Selecting this opens the Properties view for the selected item.
- When a project is selected in the explorer section, these right-click menu options are available:
  - **Scan Project**: Scan the selected application, project, or file. The scan will run with the default scan configuration.
  - **Scan Project With**: Select the scan configuration that you want to use - or choose the **Edit Configurations** action to set a different scan configuration as default (in the Scan Configuration view, select the configuration that you want to set as default, and then click **Select as Default**).
  - **Copy Project**: If a project is selected in the Explorer view, this action is available and choosing it opens a dialog box that allows you to copy the project to another application - or create a copy of the project in the application that currently contains the project.
  - **Remove Project**: Remove the selected object.
  - Register/unregister:
    - **Register Project**: Register the selected application or project with AppScan Source. You must register applications and projects before they can be published to the AppScan Source Database.
    - **Unregister Project**: Unregister the selected application or project.
    - **Locate**: Select this to associate a local application or project with one that has been registered by another AppScan Source user.
  - **Expand All**
  - **Collapse All**
  - **Properties**: Selecting this opens the Properties view for the selected item.
- When a file is selected in the explorer section, these right-click menu options are available:
  - **Scan File**: Scan the selected application, project, or file. The scan will run with the default scan configuration.
  - **Scan File With**: Select the scan configuration that you want to use - or choose the **Edit Configurations** action to set a different scan configuration as default (in the Scan Configuration view, select the configuration that you want to set as default, and then click **Select as Default**).
  - **Exclude from Scans**: Remove the selected file from scans.
  - **Open in Internal Editor**: Open the selected file in the AppScan Source editor (in the Analysis perspective).
  - **Open in External Editor**: Choose an external editor in which to open the selected file.
  - **Properties**: Selecting this opens the Properties view for the selected item.

Application and project indicators

This table identifies the application and project icons in the Explorer view.

Table 9. Application and Project Icons

Application or project type	**Not registered**	**Registered**	**Missing/Not Found**
Imported application	➜◷	➜◷▮	➜◷✗
Application that is created manually or created using the Application Discovery Assistant	◷	◷▮	◷✗
Imported project	➜☰	➜☰▮	➜☰✗
Project that is created manually or created using the Application Discovery Assistant	☰	☰▮	☰✗

The Explorer view displays local applications and projects as well as those registered on the server (those that are registered on the server but not saved locally - for example, applications and projects registered by other users - appear greyed out). If you click the toolbar **View Menu** button and toggle the **Hide items registered on the server** menu item so it is not selected, you can view existing server applications and projects. If a project is greyed out, you can right-click and choose **Locate** in the menu.

Chapter 3. Preferences

Preferences are personal choices about the appearance and operation of AppScan Source for Analysis.

To open the preference pages, select **Edit** > **Preferences** from the main workbench menu. You can browse the preferences by looking through all titles in the left pane or search a smaller set of titles by using the filter field at the top of the left pane. The results returned by the filter match both preference page titles and keywords such as JSP or email.

The arrow controls in the upper-right of the right pane enable you to navigate through previously viewed pages. To return to a page after viewing several pages, click the down arrow to display a list of your recently viewed preference pages.

General preferences

General preferences allow you to tailor some of the AppScan Source for Analysis default settings to fit your personal preferences.

Select Language

The AppScan Source for Analysis user interface can be displayed in different national languages. To change the national language that is displayed, select it from the **Select Language** list and then click **OK** in the preferences dialog box. You must manually restart the workbench for the changes to take effect.

Note: To make use of this feature, at least one language pack must be installed during the installation procedure. If English was the only language that was installed, then the product will display in English after using this preference and restarting the workbench. In this case, if you want to display a language other than English, run the installation wizard again and choose to repair the installation by adding one or more language packs.

File Encoding

The character encoding of files in your project must be set so that AppScan Source can read the files properly (and, for example, display them correctly in the source view). Select the default character encoding in this section.

Logging Level

Change the logging level to provide the level of information that you want to include in error logs. Choose from **Trace**, **Debug**, **Informational**, **Warnings**, **Errors**, and **Fatal** - where **Trace** offers the noisiest level of logging, **Fatal** logs only critical events, and the remaining settings, in order, offer incrementally higher levels of logging.

Save All Filters on Exit

When selected, you disable a prompt to automatically save all newly created or edited filters when you exit AppScan Source.

Cancel Scan on Error

When selected, incomplete scans are avoided by canceling a scan if an error occurs.

Create a marker for each finding

When selected, if you have an open assessment, source from the scan that is opened in the editor will include markers at locations for which there are findings.

By default, marker creation is enabled.

Creating markers can slow down a scan. If your project includes a many source files - or source files that are large - performance may be improved if marker creation is turned off.

After a scan finishes

The default setting is to receive a prompt asking if you want the assessment to open automatically at the end of a scan. If you do not want to the see the prompt, select **Always open the new assessment** or **Never open**.

After a scan completes and there are unsaved scans of the same targets

By default, you are prompted to save or discard the existing unsaved scans. You can modify this preference so that duplicate scans are always deleted automatically - or never deleted automatically. When setting this preference, note that memory usage is reduced when duplicate scans are deleted.

When publishing or exporting an assessment with absolute paths

By default, you are prompted to define absolute path variables when publishing. Settings in this section allow you to disable this default prompt or automatically open the dialog box that allows you to define variables when absolute paths exist.

Automatically register applications on initial publish

By default, when you publish assessments for unregistered applications or projects, you are prompted to register them. You may select to always register applications and projects when they are published - or to never register.

Important: You must have **Register** permission to be able to register applications and projects.

When application names conflict

It is possible to have applications of the same name in AppScan Source for Analysis, however, working with them can be unmanageable. By default, if you try to give an application the same name as an existing application (or import an application with the same name as an existing application), you will be prompted with a warning. The warning message allows you to generate a unique name for the application, keep the conflicting name, or cancel.

If you want to have AppScan Source for Analysis automatically generate a unique application name when conflicts arise, select **Generate Unique Name** in the preference page. If you want to automatically accept conflicting application names, select **Keep Existing Name**.

Note: Applications are stored with the filename `<application_name>.paf`. If you choose **Keep Existing Name**, you cannot set the working directory to be the same as that of the existing application with the same name. In this case, you will be prompted to overwrite the existing filename - but the overwrite will fail because the existing application is already open in AppScan Source for Analysis.

When project names conflict

This setting only applies when you attempt to create or import a project of the same name as one that exists in the same application. In this case, project names cannot conflict. If you attempt this, by default, you will be prompted to have a unique name generated - or cancel the action. If you want to have AppScan Source for Analysis automatically generate a unique project name when conflicts arise, select **Generate Unique Name** in the preference page.

Number of Assessments shown in Published Assessments and My Assessments on startup

Set the maximum number of assessments to view in the Published Assessments view or the My Assessments view.

RSS feed displayed in the Welcome View

By default, the Welcome view displays X-Force® RSS feed content. To display alternate content, enter the URL that you want in the **RSS feed displayed in the Welcome View** field.

Optimize for High Network Latency

Select this checkbox to cache more information on the client to minimize server calls.

Reload System Configuration

Load the latest system settings. If you have changed settings outside of the product while it is running - for example, by modifying `.ozsettings` files in `<data_dir>\config` (where `<data_dir>` is the location of your AppScan Source program data, as described in "Installation and user data file locations" on page 258) - select this button to refresh the settings in the product.

AppScan Enterprise Console preferences

If your AppScan Enterprise Server has been installed with the AppScan Enterprise Console option, you can publish assessments to it. The Enterprise Console offers a variety of tools for working with your assessments - such as reporting features, issue management, trend analysis, and dashboards.

To enable this feature, complete the AppScan Enterprise Console preference page. The **Domain** field in this page is optional. The remaining fields in this page must be completed with valid entries before Enterprise Console publication is enabled:

- **User ID** field: Enter the user name that you use to connect to the Enterprise Console. At a minimum, you must be a QuickScan user with your own user folder on the Enterprise Server.
- **Password** field: Enter the password used to log in to Enterprise Console (the password for the user name that was entered).
- **Domain** field: Enter the domain of the machine on which the Enterprise Console is installed.
- **Enterprise Console URL** field: Enter the URL used to access the Enterprise Console web application.

  The format of this URL is:

  `http(s)://<hostname>:<port>/ase`

  Where `<hostname>` is the name of the machine on which the Enterprise Console has been installed and `<port>` is the port on which the console is running (`<port>` does not need to be specified if the Enterprise Console is running on its default port). An example of this URL is `http://myhost.mydomain.ibm.com/ase`.

  Note:
  - This field is optional if the **Enterprise Console URL** has already been set.
  - You must be signed into AppScan Source with **Manage AppScan Enterprise Settings** permission to be able to set the **Enterprise Console URL** field. For information about user accounts and permissions, see the *Administering* section of the product infocenter - or the *Administering AppScan Source* section of the *IBM Security AppScan Source Installation and Administration Guide*.
  - The **User ID** and **Password** are stored on the machine that is running the AppScan Source client (for example, AppScan Source for Analysis) - while the **Enterprise Console URL** is stored in the Enterprise Server (which may be located on a remote machine). You cannot access user name and password information from the remote machine (for example, by issuing the `getaseinfo` command from it).
  - AppScan Source does not support publishing to an AppScan Enterprise Console instance that has been configured to use proxy settings. Attempting to publish to an instance that uses proxy settings will result in an error.

After completing the settings, it is strongly recommended that you ensure that the connection to the Enterprise Console server is valid by clicking **Test Connection**.

Tip: If the connection test fails, check that the Enterprise Console server is running and that you are able to access its control center URL in a browser (use the same **Enterprise Console URL** that you specified above).

Application server preferences for JavaServer Page compilation

If you are scanning an application that contains JavaServer Pages (JSP), the AppScan Source analysis engine must be able to compile the JSP code in order to analyze it. When you create your JSP project, you must specify the JSP compiler that AppScan Source should use (or accept the default compiler, which can be set in the Java and JSP preference page). If AppScan Source cannot compile your JSP files, use the application server preference pages to configure the JSP compiler that your application uses.

Apache Tomcat Versions 5, 6, and 7 are included in the installation of AppScan Source. If the **Tomcat 5**, **Tomcat 6**, and **Tomcat 7** preference pages are not configured, AppScan Source will compile JSP files using the supplied Tomcat JSP compiler that is currently marked as default. If you want to employ an external Tomcat compiler, use the Tomcat preference pages to point to your local Tomcat installation.

If you are using Oracle WebLogic Server or WebSphere Application Server, you must configure the applicable preference page to point to your local installation of the application server so that it can be used for JSP compilation during analysis (if you create your JSP project without first configuring the application server, you will be prompted to configure the application server at that time).

Tomcat

This topic describes the preferences that you need to set for configuring AppScan Source to reference an Apache Tomcat application server other than one that is supplied with AppScan Source.

Apache Tomcat Versions 5, 6, and 7 are included in the installation of AppScan Source. If the **Tomcat 5**, **Tomcat 6**, and **Tomcat 7** preference pages are not configured, AppScan Source will compile JSP files using the supplied Tomcat JSP compiler that is currently marked as default. If you want to employ an external Tomcat compiler, use the Tomcat preference pages to point to your local Tomcat installation.

If you are using an external supported Tomcat compiler, go to the appropriate preference page and set the application server installation directory. Specifying the installation directory allows AppScan Source to automatically find all application server dependencies when configuring projects.

WebLogic 8, 9, 11, and 12

This topic describes the preferences that you need to set for configuring AppScan Source to reference an Oracle WebLogic Server.

In both of the WebLogic preference pages, you specify the server installation directory and you can set advanced configuration options. Specifying the installation directory allows AppScan Source to automatically find all application server dependencies when configuring projects.

Configure AppScan Source to reference the WebLogic installation directory, the WebLogic JAR file, and JavaServer Page (JSP) compiler options.

Only select the **Enable Advanced Configuration Options** check box if you need to change the default WebLogic JSP compiler options or to locate the `weblogic.jar` file. The default WebLogic JSP compiler options are:

```
%JSP_JVM_OPTIONS%
 -Dcom.sun.xml.namespace.QName.useCompatibleSerialVersionUID=1.0
 -classpath
%JSP_COMPILER_CLASSPATH% weblogic.jspc
%JSP_OPTIONS% -verboseJspc -package
%PACKAGE_NAME% -linenumbers -g -debug -keepgenerated -compiler
%JAVAC_PATH% -webapp
%WEB_CONTEXT_ROOT_PATH% -d
%OUTPUT_PATH%
```

WebSphere Application Server

This topic describes the preferences that you need to set for configuring AppScan Source to referenceWebSphere Application Server for JSP compilation.

In the WebSphere Application Server preference pages, you specify the server installation directory and you can set advanced configuration options. Specifying the installation directory allows AppScan Source to find and use the WebSphere Application Server JSP compiler.

- "WebSphere Application Server 6.1"
- "WebSphere Application Server 7.0, 8.0, and 8.5"

WebSphere Application Server 6.1

Configure AppScan Source to reference the WebSphere Application Server Version 6.1 installation directory. In addition, advanced configuration options allow you to set the WebSphere Application Server JSP compiler command line and class path.

Only select the **Enable Advanced Configuration Options** check box if you need to customize the WebSphere Application Server JSP command line or to specify a class path other than the default WebSphere Application Server class path (alter this setting if you would like to include additional jars in the class path used by all your WebSphere Application Server Version 6.1 applications).

The default WebSphere Application Server JSP compiler command line option is:

```
%CMD_EXE% %CMD_ARGS%
'%FILE(%%JSP_COMPILER_INSTALL_DIR%/bin/JspBatchCompiler%BAT%%)%'
-response.file
'%TMP_FILE(%-keepgenerated=true -recurse=true -useFullPackageNames=true
-verbose=false -createDebugClassfiles=true -jsp.file.extensions=%WEB_EXTS%
-javaEncoding=%ENCODING%
%JSP_OPTIONS% %QUOTE%-war.path=%WEB_CONTEXT_ROOT_PATH%%QUOTE%
%QUOTE%-filename=%RELATIVE_FILENAME_NO_QUOTE% %QUOTE%  %)%'
```

WebSphere Application Server 7.0, 8.0, and 8.5

Configure AppScan Source to reference the WebSphere Application Server installation directory. In addition, advanced configuration options allow you to set the WebSphere Application Server JSP compiler command line and class path.

Only select the **Enable Advanced Configuration Options** check box to customize the WebSphere Application Server JSP command line or to specify a class path other than the default WebSphere Application Server class path (alter this setting if you would like to include additional jars in the class path used by all your WebSphere Application Server applications).

The default WebSphere Application Server JSP compiler command line option is:

```
%CMD_EXE% %CMD_ARGS%
'%FILE(%%JSP_COMPILER_INSTALL_DIR%/bin/JspBatchCompiler%BAT%%)%'
-response.file
'%TMP_FILE(%-keepgenerated=true -recurse=true -useFullPackageNames=true
-verbose=false -createDebugClassfiles=true -jsp.file.extensions=%WEB_EXTS%
-javaEncoding=%ENCODING%
%JSP_OPTIONS% %QUOTE%-war.path=%WEB_CONTEXT_ROOT_PATH%%QUOTE%
%QUOTE%-filename=%RELATIVE_FILENAME_NO_QUOTE% %QUOTE%  %)%'
```

Defining variables

When saving assessments or bundles, or publishing assessments, AppScan Source for Analysis may suggest that you create a variable to replace absolute paths (without variables, AppScan Source for Analysis writes absolute paths to the assessment file to reference items such as source files). When you configure variables for absolute paths, you facilitate the sharing of assessments on multiple computers. It is recommended that you use variables when sharing assessments.

About this task

Variables can be created prior to initiating a save or publish action by following the instructions in this topic - or they can be created after initiating the save or publish action by following the steps in “Defining variables when publishing and saving” on page 96.

To learn, by example, how variables can assist when sharing assessments, see “Example: Defining variables” on page 96.

Procedure

1. Select **Edit** > **Preferences** from the main menu. In the Preferences dialog box, choose **Change Variables**.
2. Click the **Add Variable** button in the Change Variables preference page.
3. Enter a name for the variable and browse to the file location that will be replaced by the variable (AppScan Source for Analysis inserts the surrounding percent symbols (%) after the variable is created).
4. Repeat the above step for any other reference items in the assessment (for example, if the assessment references source in multiple locations, add a variable for each location).
5. Use the preference page to edit and remove variables, using the **Modify Variable** and **Delete Variable** buttons.
6. Click **OK** when you are finished defining variables.

Enabling defect tracking with preferences

Defect Tracking System preferences allow you to enable the submission of findings to a defect tracking system - and determine how defects are submitted.

The General tab in the Defect Tracking System preference page is used to enable or disable the Defect Tracking System integration feature in AppScan Source. If the **Enable Defect Tracking System Integration** checkbox is selected, the **Submit Defect** context menu action will be available for assessment findings. The General tab also provides discrete control over which Defect Tracking Systems will be available when submitting defects.

To learn about the preferences that can be set for supported defect tracking systems, refer to these help topics:
- “Rational ClearQuest preferences”
- “Quality Center preferences” on page 70
- “Rational Team Concert preferences” on page 72
- “Team Foundation Server preferences” on page 74

Rational ClearQuest preferences

To be able to complete Rational ClearQuest preferences, your Rational ClearQuest administrator must provide you with the required Rational ClearQuest settings. The settings are specific to your Rational ClearQuest environment.

Note: When integrating with Rational ClearQuest Version 8.0, the Rational ClearQuest schema must contain the fields that are available in the **DefectTracking** predefined schema.

Database set

A collection of one or more defect databases.

```
Linux default = Connection Name,
Windows default = Database Set
```

Database name

Name of the database to which to submit defects.

Database user name

Default Rational ClearQuest database user name.

Location of CQPerl executable

Location of the Rational ClearQuest CQPerl executable on the local computer. The provided default location maps to the default Rational ClearQuest installation location.

Entity for defect records

Entity (database object) configured by the Rational ClearQuest installation for use for defect objects.

The default entity is **Defect**.

Description field on record

Default Description is **Description**.

Headline field on record

Default Headline is **Headline**.

Single defect per finding

Submit a group of findings as a single defect or as multiple defects. You can change the submission method when you create the defect.

Quality Center preferences

You must first enable HP Quality Center as a General Defect Tracking System preference and then set the individual preference on the Quality Center tab.

Server URL

The Quality Center Server URL - for example, `http://<hostname>:<port>/qcbin/` or `https://<hostname>:<port>/qcbin/`.

User Name (Optional)

A user name to log in to Quality Center

Password (Optional)

If you entered a user name, enter the password for it

Domain

The Quality Center domain to which to connect.

Project

The Quality Center project to which to connect

Auto-login

When true, AppScan Source does not prompt for login information when submitting findings, and logs in with the default credentials specified in the Preferences. When false, you must log in each time you submit a finding to Quality Center.

Auto-submit

When true, the dialog box that is used for submitting new defects does not appear when submitting findings. AppScan Source for Analysis uses the **Default Defect Properties** specified in the Preferences. When false, a prompt appears requesting that you enter defect information (Severity, Priority, Defect Type, Status, and so forth) when submitting findings.

Resubmit previously-submitted findings

Findings submitted to Quality Center are tagged with Quality Center defect information (Defect ID, submitting user, and submission date). By default, AppScan Source does not resubmit the same finding more than once. This allows you to dispatch multiple findings to Quality Center, only entering new findings in the Quality Center database. When selected (true), previously submitted findings can be resubmitted to Quality Center.

Submit each finding as an individual bug

When submitting multiple findings in a single operation, you can either submit all findings as a single Quality Center defect or as a separate Quality Center defect for each individual AppScan Source finding. Selecting this check box sets the flag to true, creating a separate Quality Center defect for each individual finding. Setting the flag to false creates one Quality Center defect for all findings submitted as part of a bulk submission.

Auto Generate Bug Summary

When true, AppScan Source automatically generates a defect summary for the submission in Quality Center. The summary indicates the number of findings included in the defect and the type of findings included, such as `Validation.Required`.

When false, the Summary field appears for you to complete when submitting the defect in the dialog box that opens when you create a new defect.

Auto Load Bug Fields

Default setting is true. When the check box is selected, AppScan Source automatically loads defect field definitions from the Quality Center database, based on the current user and group settings in Quality Center. When false, AppScan Source does not display defect fields from Quality Center in the dialog box that opens when you create a new defect.

Default Defect Properties

To set default values for the different Quality Center defect attributes, click **Default Defect Properties** on the Quality Center preference tab.The default values either pre-populate the **New Defect** dialog box at submission time, or they are sent to Quality Center silently if the **Auto-submit** preference is selected.

Note: Defect properties and their available values are pulled dynamically from Quality Center each time the **Defect Properties** dialog box appears if **Auto Load Bug Fields** is selected. Therefore, any new fields and values added to the Quality Center database automatically appear in AppScan Source for Analysis. Valid server, login, and connection information is required to open and populate the **Defect Properties** dialog box with Quality Center information.

Customizing Quality Center defect fields

Through a configuration file, you can customize the fields and interactions between these fields in the New Defect dialog box. You can find an example configuration file in `<data_dir>\config\qc.dts` (where `<data_dir>` is the location of your AppScan Source program data, as described in "Installation and user data file locations" on page 258), which contains sample customizations and additional documentation. These customizations allow you to model your Quality Center Workflow script logic directly in the New Defect dialog box.

Available customizations include:
- Displaying custom fields, missing fields, or both
- Forcing fields to always display (overriding Quality Center settings)
- Updating required state of fields based on selection of other fields
- Dynamically updating list box options for a field based on the list box selection in another field

Rational Team Concert preferences

The Rational Team Concert preference tab allows you to configure a connection to a Rational Team Concert server and also to configure the values of work item attributes.

Once you have entered your connection information and successfully logged in, you can then choose to connect to one or more project areas. Each project area can have its own configuration of attribute preset values.

Note: When you connect to Rational Team Concert (by configuring preferences or submitting defects), you may be prompted to accept an SSL certificate. See "Rational Team Concert SSL certificates" on page 73 for more information.

To configure the attribute values for a given project area, select the project area and choose **Configure**. In the configuration dialog box, you can set attribute values to either hardcoded values or in some cases to variables that refer to a selected finding. For example, the use of `{Finding.fileName}` in an attribute value will be replaced with the actual source code file name for a finding during submission. Content Assist (`<Ctrl>+<Space>`) is provided for attribute values that support these variables. Teams are encouraged to share these configurations using the **Import** and **Export** buttons that are available on the main Rational Team Concert preference page.

Resolving Rational Team Concert server mismatches

AppScan Source can interact with only one Rational Team Concert server version at a time - either Version 2.x or Version 3.x.

Procedure

1. To toggle the Rational Team Concert version, perform one of these actions:
   - To enable Rational Team Concert Version 2.x, copy the `<install_dir>\configuration\rtc2config\config.ini` file up one level to `<install_dir>\configuration` (where `<install_dir>` is the location of your AppScan Source installation).
   - To enable Rational Team Concert Version 3.x, copy the `<install_dir>\configuration\rtc3config\config.ini` file up one level to `<install_dir>\configuration`.
2. Restart AppScan Source.

   Note: It may also be necessary to delete your `<user_home>\.ounceconfiguration` directory before restarting AppScan Source for this change to take effect (where `<user_home>` is your operating system home directory (for example, on Windows, the directory might be `C:\Documents and Settings\Administrator\`)).

Rational Team Concert SSL certificates

When a Rational Team Concert server is installed, it should be configured to use a valid SSL certificate. If this is not done, you will receive an untrusted connection message when logging in to the server (while configuring preferences or submitting defects). This topic outlines Rational Team Concert SSL certificate considerations.

SSL certificate storage location

Certificates that have been permanently accepted are stored in `<user_home>/.jazzcerts` (where `<user_home>` is your operating system home directory (for example, on Windows, the directory might be `C:\Documents and Settings\Administrator\`)). Removing `<user_home>/.jazzcerts` deletes all stored certificates for AppScan Source and Rational Team Concert clients.

SSL certificate sharing with Rational Team Concert clients

AppScan Source shares its certificate store with Rational Team Concert clients. If you permanently accept a certificate using a Rational Team Concert client, it will be reused by AppScan Source (you will not be prompted in AppScan Source to accept a certificate). Similarly, if you permanently accept a certificate in AppScan Source, it will be reused by Rational Team Concert clients.

Rational Team Concert server rename considerations when defect tracking from AppScan Source for Analysis

If you have enabled Rational Team Concert defect tracking in AppScan Source for Analysis, and the Rational Team Concert server is renamed, any existing configuration for project areas on that server will no longer be available in AppScan Source for Analysis. You will need to connect to the server via its new repository URI and re-create the configuration in the Defect Tracking System preferences.

Team Foundation Server preferences

The Team Foundation Server preference tab allows you to configure a connection to a Microsoft Team Foundation Server and to configure the values of work item fields.

Once you have entered your connection information and successfully logged in, you can then choose to connect to one or more projects.

Note: When configuring the login to Team Foundation Server 2010, the Server URL must contain the Team Project Collection you want to connect to. For example, `http://myserver:8080/tfs/`*`DefaultCollection`*.

Each project can have its own configuration of field preset values.

To configure the field values for a given project, select the project and choose **Configure**. In the configuration dialog box, you can set field values to either hardcoded values or in some cases to variables that refer to a selected finding. For example, the use of `{Finding.fileName}` in a field value will be replaced with the actual source code file name for a finding during submission. Content Assist (`<Ctrl>+<Space>`) is provided for fields that support these variables.

Teams are encouraged to share these configurations using the **Import** and **Export** buttons that are available on the main Team Foundation Server preference page.

Eclipse workspace importers: Eclipse or Rational Application Developer for WebSphere Software (RAD) preference configuration

The AppScan Source for Analysis installation provides a default Eclipse importer. This importer identifies the location of Eclipse and the JRE. If the default Eclipse importer is unable to import your workspace, it may be necessary to create a new Eclipse importer.

Before you begin

Each importer configuration represents an installation of Eclipse or Rational Application Developer for WebSphere Software (RAD). To use these configurations to import existing workspaces and projects to AppScan Source for Analysis, you may also need to install the AppScan Source for Development plug-ins into the Eclipse environment.

Before adding a RAD workspace, you must create a configuration for the workspace type.

Procedure

1. In AppScan Source for Analysis, select **Edit** > **Preferences** from the main workbench menu.

2. Select **Eclipse Workspace Importers**.
3. Click **Create a new configuration** and then complete the New Import Configuration dialog box to create a new configuration:
   - **Product**: Select the appropriate product

     Note: If the product that was used to create the workspace is not available for selection, ensure that you have completed the configuration steps outlined in "Eclipse or Application Developer updates" on page 26 before attempting to create the workspace importer.
   - **Name**: Importer name
   - **Location**: Path to the base directory of the Eclipse installation
   - **JRE Location**: Path to the root directory of the Java Runtime Environment (JRE). Use a JDK in `<install_dir>\JDKS` (where `<install_dir>` is the location of your AppScan Source installation) or any other preferred JDK.
4. Click **OK**.
5. To identify the importer as default, select it and click **Make the selected configuration the default**. This causes an icon to display in the importer's **Default** column.

Email

Configure email settings used to send findings under consideration as defects.

- **To Address**: Recipient's email address. By default, the **Mail To** field in the Email Findings dialog box will populate with this email address - however it can easily be changed when preparing the email.
- **From Address**: Sender's email address.

  Note: A valid email address is recommended to avoid having the recipient mail client treat the email as spam.
- **Mail Server**: SMTP mail server configured as `mail.myexample.com`.

  Important: Check with your system administrator to ensure that you have the correct mail server information.

Java and JavaServer Pages

Use this preference page to add, modify, or delete the Java Development Kit (JDK) that is used for scans (and to set a default JDK). In addition, use the page to set the default JavaServer Page (JSP) compiler.

Default

Identifies the location of the JDK used for scans. A scan uses the default JDK path when the project does not specify an explicit JDK. To set the JDK as the default, right-click the identified JDK name and click **Set Default JDK**. The default icon appears in the table, identifying the current default JDK.

Note: The default compiler for JSP projects is Tomcat 7 (Jasper 2), which requires Java Version 1.6 or higher. If **Tomcat 7** is kept as default, selecting an earlier JDK (for example, **IBM JDK 1.5**) will result in compilation errors during scans.

JDK Name and Path

Identifies the name and location of the JDK.

Default Compiler for JSP Projects

Tomcat 7 (Jasper 2) is the default JSP compiler setting. To learn about the compilers that are supported by AppScan Source for Analysis, see http://www.ibm.com/support/docview.wss?uid=swg27027486.

Knowledgebase articles

Use the Knowledgebase Articles preference page to set locations that contain AppScan Source Security Knowledgebase articles.

The page lists the directories that contain articles. To add a directory, click **Add Content Directory** and then browse to the article location. To remove a directory, select it and then click **Remove**.

Project file extensions

Configure or add valid global file extensions for each project type, change the extensions to include in scans, exclude from scans, and specify extensions as web files.

A tab page appears for each available language or project type: Java, JavaScript, ASP, Perl, PHP, ColdFusion, PBSA (for the pattern-based project type), COBOL, PL/SQL, T-SQL, VB.NET, .Net Assembly, VB, C/C++, ASP .NET 1.x, ASP .NET 2.x, WSDL, and C#. When adding a new extension, identify if files with the new extension can be scanned, considered as a web file, or excluded.

Settings in this page are global. To set file extensions for individual projects, use the “File Extensions” on page 196 tab of the Properties view for the selected project.

File extension settings

Table 10. File extension settings

Setting	Description	Usage examples
Scan or **Assess**	Include files with the indicated extension in full analysis.	• If a `.xxx` extension is created for Java projects and marked as **Scan** or **Assess**, files with that extension will be compiled and scanned. • A file can be part of a project but not marked as **Scan** or **Assess** if it should not be compiled and scanned (such as header files in C++). These files would be included in the project and searched during pattern-based analysis.

Table 10. File extension settings (continued)

Setting	Description	Usage examples
Web File	Mark files with the indicated extension for JSP compilation. This setting allows AppScan Source to separate web sources from non-web sources.	If a `.yyy` extension is created for Java projects and marked as a **Web File**, files with that extension will be arranged as web sources in the projects. When AppScan Source prepares for analysis, these files will be precompiled into classes to be analyzed.
Exclude	Do not create source files in the project for files with the indicated extension. Files with this extension will not be scanned.	Create a `.zzz` extension for files that are necessary to your projects for compilation, but do not need to be included in analysis.

Chapter 4. Scanning source code and managing assessments

This section explains how to scan your source code and manage assessments.

Once you configure applications and projects - or use the Application Discovery Assistant to create applications and projects for you - you are ready to scan the source code. The result of a scan, an *assessment*, can be saved or published. A *saved assessment* is a file of scan results saved locally that can be published, opened later for additional triage, or opened in AppScan Source for Development. A *published assessment* consists of the scan results saved to the AppScan Enterprise Server.

You manage assessments in two views:
- My Assessments
- Published Assessments

Note: When saving, publishing, or opening an assessment, progress appears in the status bar.

Scanning source code

This task describes the various methods for launching scans.

About this task

You can scan at various levels (all applications, one or more applications, one or more projects, or one or more files). If you have completed a scan, you can scan it again, provided its assessment is open.
- "Scanning all applications" on page 80
- "Scanning one or more applications" on page 80
- "Scanning one or more projects" on page 80
- "Scanning one or more files" on page 81
- "Re-scanning code" on page 81

A scan configuration is always used when scanning. To learn more about scan configurations, see "Managing scan configurations" on page 81 and the scan option descriptions below.

Restriction: When you scan multiple applications or projects, a parent node containing assessments for each scanned item is created in the My Assessments view. The individual child assessments cannot be managed in this case (for example, the child assessments cannot be removed or published individually). When multiple applications or projects are scanned at the same time, you can only manage the assessments as a group (the parent node).

Note:
- If you are scanning an IBM Worklight project that contains files that have been modified, you must rebuild the project using IBM Worklight and then refresh your AppScan Source application or project before scanning. If you do not rebuild the project using IBM Worklight, modifications made to files will be ignored by AppScan Source.

- The AppScan Source for Analysis client is built on Eclipse. On Linux, Eclipse requires the installation of a third-party component in order to render browser-based content. Without this component, AppScan Source for Analysis may exhibit symptoms such as a hang after login or a fail during product use. See "Enabling browser-based content on Linux for AppScan Source for Analysis" on page 87 for more information.

Scanning all applications

Procedure

Complete one of these actions:

1. Select **Scan** > **Scan All** in the main workbench menu. The scan will run with the default scan configuration.
2. In the Explorer view:
   - Right-click **All Applications** and then select **Scan All Applications** from the menu. The scan will run with the default scan configuration.
   - To use a different scan configuration for the scan, right-click **All Applications** and then select **Scan All Applications With** from the menu. Select the scan configuration that you want to use - or, to set a different default scan configuration, choose the **Edit Configurations** action (in the Scan Configuration view, select the configuration that you want to set as default, and then click **Select as Default**).

Scanning one or more applications

Procedure

1. In the Explorer view, select one or more applications.
2. Complete one of these actions:
   a. Select **Scan** > **Scan Selection** in the main workbench menu. The scan will run with the default scan configuration.
   b. In the Explorer view:
      - Right-click the selection and choose **Scan Application** from the menu. The scan will run with the default scan configuration.
      - To use a different scan configuration for the scan, right-click the selection and then select **Scan Application With** from the menu. Select the scan configuration that you want to use - or, to set a different default scan configuration, choose the **Edit Configurations** action (in the Scan Configuration view, select the configuration that you want to set as default, and then click **Select as Default**).

Scanning one or more projects

Procedure

1. In the Explorer view, select one or more projects.
2. Complete one of these actions:
   a. Select **Scan** > **Scan Selection** from the main workbench menu. The scan will run with the default scan configuration.
   b. In the Explorer view:
      - Right-click the selection and choose **Scan Project** from the menu. The scan will run with the default scan configuration.
      - To use a different scan configuration for the scan, right-click the selection and then select **Scan Project With** from the menu. Select the scan

configuration that you want to use - or, to set a different default scan configuration, choose the **Edit Configurations** action (in the Scan Configuration view, select the configuration that you want to set as default, and then click **Select as Default**).

Scanning one or more files

About this task

The ability to scan individual or multi-selected files is not available for Microsoft .NET files (for example, `.cs` and `.vbnet`).

Procedure

1. In the Explorer view, select one or more files.
2. Complete one of these actions:
   a. Select **Scan** > **Scan Selection** from the main workbench menu. The scan will run with the default scan configuration.
   b. In the Explorer view:
      - Right-click the selection and choose **Scan File** from the menu. The scan will run with the default scan configuration.
      - To use a different scan configuration for the scan, right-click the selection and then select **Scan File With** from the menu. Select the scan configuration that you want to use - or, to set a different default scan configuration, choose the **Edit Configurations** action (in the Scan Configuration view, select the configuration that you want to set as default, and then click **Select as Default**).

Re-scanning code

Procedure

To re-scan the current target, select **Scan** > **Scan Again** from the main menu. The last scan configuration that was used to scan the item (or selected items) will be used again for the scan:

- If the default scan configuration was used for the previous scan and if a new default scan has been set, the new default scan configuration will be used for the scan.
- If a non-default scan configuration was used for the previous scan, it will be used for the scan. If that scan configuration has been modified and saved since the previous scan, the modified scan configuration will be used.

Managing scan configurations

Scan configurations are used when launching scans. In a scan configuration, you can specify source rules to use during a scan. The settings made in a scan configuration can often lead to better scan results - and the ability to save these settings can make scanning easier and more time-efficient.

About this task

This task describes the steps involved in managing scan configurations.

- "Creating a scan configuration" on page 82
- "Modifying a scan configuration" on page 83
- "Removing a scan configuration" on page 83

- "Sharing scan configurations and working with shared configurations" on page 83
- "Setting scan configurations as default" on page 83
- "Built-in scan configurations" on page 84

Note: Scan configurations do not apply to quality scans or when scanning JavaScript, ColdFusion, Perl, Cobol, PL/SQL, or T-SQL.

Scan configurations are managed in the "Scan Configuration view" on page 84. This view can be opened by selecting **View** > **Scan Configuration** from the main menu bar - or by selecting the **Edit Configurations** action in the Explorer view.

Once you have scan configurations in place, you can use them when launching scans in AppScan Source for Analysis (refer to "Scanning source code" on page 79 for more information). You can also use scan configurations when launching scans in AppScan Source for Automation, AppScan Source for Development, and the AppScan Source command line interface (CLI).

Creating a scan configuration

Procedure

1. Complete one of these actions:
   a. Click the Scan Configuration view **New** button.
   b. Select an existing configuration from the list and then click **Duplicate**. This will cause a scan configuration to be created based on the original scan configuration's settings - which you can modify and save as a new configuration.
2. In the General section, enter a unique name for the configuration in the **Name** field.
3. Optional: Enter a description for the scan configuration.
4. Optional: The Scan Rules section allows you to specify the source rules that will be in effect for the scan (see "Scan Rules" on page 85 for more information). In this section, you can choose to scan with selected source rule sets - or you can select individual rule properties to use for the scan:
   a. By default, this section allows you to choose rule sets to apply. Select one or more of the available rule set check boxes.
   b. To choose individual rule properties instead of rule sets, click **Discard selected rule sets and let me select individual rule properties**. This opens the Select Rule Properties dialog box, which allows you to choose individual rule properties. If this dialog box is completed, any rule sets that were selected will be discarded. Scan rules that have the selected rule properties will be used for the scan.

   If you choose individual rule properties for the scan - and want to select rule sets instead, click **Discard selected rule properties and let me select by rule set**. This will discard any rule properties that had been selected in the Select Rule Properties dialog box, and allow you to select rule sets instead.

   Note: When selecting individual rule properties, the selected items apply to the properties of sources, not sinks. This means that you are limiting the attack surface to only sources that have the selected properties. You may see vulnerabilities in your results that do not match the selected properties, since vulnerability types are based on the sink and not the source.

5. Optional: The Advanced Settings section is intended for advanced users only. It contains a variety of settings that can improve scan results. Hover text describes each setting in this section.

   Note:
6. Once all settings have been made in the scan configuration, click **Save**.

Modifying a scan configuration

Procedure

1. In the Scan Configuration view, select the scan configuration that you want to modify.

   Note: To share scan configurations - or modify or delete a shared scan configuration - you must have must have **Manage Shared Configurations** permission. To learn about setting permissions, see the *IBM Security AppScan Source Installation and Administration Guide*.
2. After modifying the scan configuration, click **Save**.

Removing a scan configuration

Procedure

1. In the Scan Configuration view, select the scan configuration that you want to remove.
2. Click **Delete**.

Sharing scan configurations and working with shared configurations

About this task

Scan configurations can be saved to the AppScan Source Database for the purpose of sharing them with others. To share a scan configuration with others, click **Share**.

Note: To share scan configurations - or modify or delete a shared scan configuration - you must have must have **Manage Shared Configurations** permission. To learn about setting permissions, see the *IBM Security AppScan Source Installation and Administration Guide*.

Scan configurations that have been shared by others appear in the list of scan configurations.

Note: Once a scan configuration has been shared, you cannot remove the share. You can complete one of these tasks instead:
- Delete the shared scan configuration. This will delete it locally and on the server.
- Duplicate the shared scan configuration and then delete it. Duplicating the scan configuration will create an identical local copy of it.

Setting scan configurations as default

About this task

You can set any scan configuration to be default - whether it be local, built-in, or shared. If you set a shared scan configuration as default, the setting is only made locally and does not affect other users.

To learn how the default scan configuration is used, refer to "Scanning source code" on page 79.

Procedure

1. In the Scan Configuration view, select the scan configuration that you want to set as default.
2. Click **Select as Default**.

Built-in scan configurations

About this task

AppScan Source provides built-in scan configurations. These cannot be modified or removed. Selecting them in the list will allow you to duplicate them or view their settings.

Scan Configuration view

The Scan Configuration view allows you to create configurations that you can use when launching scans - and you can set the default scan configuration in the view. In a scan configuration, you can specify source rules to use during a scan - and you can include numerous scan settings. The settings made in a scan configuration can often lead to better scan results - and the ability to save these settings can make scanning easier and more time-efficient.

The Scan Configuration view has four main sections:
- "Scan configuration management"
- "General" on page 85
- "Scan Rules" on page 85
- "Advanced Settings" on page 86

Note: Scan configurations do not apply to quality scans or when scanning JavaScript, ColdFusion, Perl, Cobol, PL/SQL, or T-SQL.

Scan configuration management

Use this section to select, add, remove, save, and share scan configurations - and to set scan configurations as default.
- To create a new scan configuration, click **New**. After completing the scan configuration settings, click **Save** to save the changes. To set the scan configuration as default, click **Select as Default** after saving it. To learn how the default scan configuration is used, refer to "Scanning source code" on page 79.
- To work with an existing scan configuration, select it from the list:
  - If you modify the scan configuration settings, click **Save** to save the changes (unwanted changes can be discarded by switching to a different scan configuration and then clicking **Discard**).
  - To remove the selected scan configuration, click **Delete**.
  - To duplicate the scan configuration, click **Duplicate**. This will cause a new scan configuration to be created based on the original scan configuration's settings.
  - To set the scan configuration as default, click **Select as Default**. To learn how the default scan configuration is used, refer to "Scanning source code" on page 79.
  - To share the scan configuration with others, click **Share**. This will save the scan configuration to the AppScan Enterprise Server.

    Note: To share scan configurations - or modify or delete a shared scan configuration - you must have must have **Manage Shared Configurations**

permission. To learn about setting permissions, see the *IBM Security AppScan Source Installation and Administration Guide*.

Note: AppScan Source provides built-in scan configurations. These cannot be modified or removed. Selecting them in the list will allow you to duplicate them or view their settings.

General

This section allows you to name scan configurations and provide descriptions for them.

Scan Rules

Use this section to determine which source rules will be in effect for the scan.

A source is an input to the program, such as a file, servlet request, console input, or socket. By excluding some source rules, you can speed up scanning and avoid detecting vulnerabilities arising from inputs that are not of interest.

Rules are tagged with rule properties to indicate that they are related to a particular vulnerability, mechanism, attribute, or technology. These properties are grouped into rule sets, which correspond to a common set of related rules. You can limit the source rules included in the scan by specifying either rule sets or individual rule properties.

- Select one or more vulnerability types (organized by type in rule sets) to include in the scan:
  - **Everything**: If this is selected, vulnerabilities arising from all supported sources of input will be detected.
  - **User Input**: If this is selected, vulnerabilities arising from end user input will be detected.
  - **Web Applications**: If this is selected, vulnerabilities arising from web application risk will be detected.
  - **Error Handling and Logging**: If this is selected, vulnerabilities arising from error handling and logging mechanisms will be detected.
  - **Environment**: If this is selected, vulnerabilities arising from configuration files, system environment files, and property files will be detected.
  - **External Systems**: If this is selected, vulnerabilities arising from external entities will be detected.
  - **Data Store**: If this is selected, vulnerabilities arising from data stores (such as databases and caching) will be detected.
  - **Unusual Things**: If this is selected, vulnerabilities arising from routines that are not normally part of a production application will be detected.
  - **File System**: If this is selected, vulnerabilities arising from file systems will be detected.
  - **Sensitive Data**: If this is selected, vulnerabilities arising from sensitive data will be detected.

  Hover text describes each rule set in this section.
- Select individual scan rule properties to include in the scan: Click **Discard selected rule sets and let me select individual rule properties**. This opens the Select Rule Properties dialog box, which allows you to choose individual rule

properties. If this dialog box is completed, any rule sets that were selected will be discarded. Scan rules that have the selected rule properties will be used for the scan.

Advanced Settings

This section is intended for advanced users only. It contains a variety of settings that can improve scan results. Hover text describes each setting in this section.

Excluding a file from a scan

Before you begin

Note: If your application is an Eclipse workspace, you cannot exclude files from scans.

Procedure

1. In the Explorer view, select the file or files to eliminate from the scan.
2. Right-click the selection and choose **Exclude from Scans** from the menu.

Results

When the Properties view is opened for the project that contains the file, the view's **Sources** tab lists files in the project, including excluded files.

Project files are listed under the **Source Root** icon. Files that are excluded from scanning have a red file icon (if an excluded file is right-clicked, its menu has **Exclude** disabled and **Include** enabled). To exclude an included file, right-click it and choose **Exclude** in the menu. To include an excluded file, right-click it and choose **Include** in the menu.

Cancelling or stopping a scan

Although you can cancel a scan in progress, canceling a scan causes a loss of all data for that scan. Alternatively, you can stop a scan to halt it and produce an assessment with results that are found so far.

Cancelling or stopping a scan in AppScan Source for Analysis

To cancel a scan that is currently in progress, select **Scan** > **Cancel Scan** or **Scan** > **Stop Scan** from the main menu.

Cancel Scan terminates the scan and does not produce any results. **Stop Scan** halts the scan and produces an assessment with results found so far.

Cancelling or stopping a scan in AppScan Source for Development (Eclipse plug-in)

While the scan is running:

- To cancel the scan, choose **Security Analysis** > **Scan** > **Cancel Scan** from the main menu. The scan terminates and does not produce any results - and cancellation diagnostic messages appear in the Eclipse Console.
- To stop the scan, choose **Security Analysis** > **Scan** > **Stop Scan** from the main menu. The scan terminates and produces an assessment of the results that were collected until the stop action was initiated.

Note: The AppScan Source for Development (Eclipse plug-in) is supported on Windows and Linux only.

Cancelling a scan in AppScan Source for Development (Microsoft Visual Studio plug-in)

While the scan is running, choose **IBM Security AppScan Source** > **Scan** > **Cancel Scan** from the main menu. The scan terminates and does not produce any results.

Note: The AppScan Source for Development Microsoft Visual Studio plug-in is supported on Windows only.

AppScan Source for Analysis and AppScan Source for Development (Eclipse plug-in) component prerequisite on Linux

On Linux, Eclipse requires the installation of a third-party component in order to render browser-based content. Without this component, AppScan Source for Analysis and the AppScan Source for Development Eclipse plug-in may exhibit symptoms such as a hang after login or a fail during product use.

Information about this prerequisite is available at http://www.eclipse.org/swt/faq.php#browserwebkitgtk.

- "Enabling browser-based content on Linux for AppScan Source for Analysis"
- "Enabling browser-based content on Linux for AppScan Source for Development installed to Eclipse Version 3.7 or later"
- "Enabling browser-based content on Linux for AppScan Source for Development installed to Eclipse Version 3.6 or earlier" on page 88

Enabling browser-based content on Linux for AppScan Source for Analysis

AppScan Source for Analysis is built on Eclipse and is, therefore, affected by this issue.

The recommended approach for correcting this is to ensure that a 32-bit or i686 version of WebKitGTK 1.2.0 or later is installed. You should consult with your system administrator for the proper way to get packages installed, but on some systems this may be as simple as issuing `yum install webkitgtk.i686`.

If you are unable to install WebKitGTK, you can choose to install a 32-bit version of Mozilla XULRunner 1.8. With this option, you may also need to make these updates to your environment variables:

- Set `MOZILLA_FIVE_HOME` to the XULRunner installation location.
- Update `LD_LIBRARY_PATH` to append (or pre-pend) `$MOZILLA_FIVE_HOME`

Enabling browser-based content on Linux for AppScan Source for Development installed to Eclipse Version 3.7 or later

The recommended approach for correcting this is to ensure that a 32-bit or i686 version of WebKitGTK 1.2.0 or later is installed. You should consult with your system administrator for the proper way to get packages installed, but on some systems this may be as simple as issuing `yum install webkitgtk.i686`.

If you are unable to install WebKitGTK, you can choose to install a 32-bit version of Mozilla XULRunner 1.8. With this option, you may also need to make these updates to your environment variables:

- Set `MOZILLA_FIVE_HOME` to the XULRunner installation location.
- Update `LD_LIBRARY_PATH` to append (or pre-pend) `$MOZILLA_FIVE_HOME`

Enabling browser-based content on Linux for AppScan Source for Development installed to Eclipse Version 3.6 or earlier

Ensure that you have a 32-bit version of Mozilla XULRunner Version 1.8 installed (Version 1.8.0.4 works in most environments - see https://developer.mozilla.org/en-US/docs/XULRunner_1.8.0.4_Release_Notes). After installing XULRunner, you may also need to make these updates to your environment variables:

- Set `MOZILLA_FIVE_HOME` to the XULRunner installation location.
- Update `LD_LIBRARY_PATH` to append (or pre-pend) `$MOZILLA_FIVE_HOME`

Managing My Assessments

The My Assessments view contains a list of assessments (the currently-opened assessment, along with any assessments that you have saved). In this view, you can open, delete, save, rename, or compare assessments. When a scan completes or you open a saved assessment, the assessment appears in the My Assessments view. My Assessments displays a table of open or saved assessments, and identifies a published or modified assessment. Removing an assessment from this view (without saving or publishing it) permanently deletes that assessment.

For more information about the My Assessments view, see "My Assessments view" on page 238.

Restriction: When you scan multiple applications or projects, a parent node containing assessments for each scanned item is created in the My Assessments view. The individual child assessments cannot be managed in this case (for example, the child assessments cannot be removed or published individually). When multiple applications or projects are scanned at the same time, you can only manage the assessments as a group (the parent node).

Tip: You can only open scan results that pertain to one application at a time. To view results of a multi-application or multi-project scan, you must expand the tree in the My Assessments view and double-click the assessment that you want to open.

Publishing assessments

AppScan Source offers two publishing options. You can publish assessments to the AppScan Source Database, for the purpose of storing and sharing assessments. Or, if your AppScan Enterprise Server has been installed with the Enterprise Console option, you can publish assessments to it. The AppScan Enterprise Console offers a variety of tools for working with your assessments - such as reporting features, issue management, trend analysis, and dashboards.

To learn more about AppScan Source publishing features, see "Publishing assessments to AppScan Source" on page 89 and "Publishing assessments to the AppScan Enterprise Console" on page 91.

Note: When publishing to the AppScan Enterprise Console, the version and release level of AppScan Source must match that of the AppScan Enterprise Server. For example, you can publish from AppScan Source Version 8.6.x to AppScan Enterprise Server Version 8.6.x - but you cannot publish from AppScan Source Version 8.6.x to AppScan Enterprise Server Version 8.5.x.

Registering applications and projects for publishing to AppScan Source

Before an assessment can be published to the AppScan Source Database, the applications or projects that were scanned in order to create it must be registered. By default, if you attempt to publish an assessment of unregistered applications or projects, you will be prompted to register the applications or projects at that time. AppScan Source for Analysis *auto-registers* for you if your **General** preference, **Automatically register applications on initial publish**, is **Always register**.

Important: You must have **Register** permission to be able to register applications and projects.

To register applications and projects before scanning, select the applications or projects in the Explorer view, and then select **File** > **Register** from the main workbench menu. **Register Application** and **Register Project** actions are also available when you right-click the selected items in the Explorer view.

If an application is already registered, you can register it again by a new name. To do this, select and right click it and then choose **Register Application As** from the menu. In the Rename dialog box, enter a new name for the registered application or project.

To unregister applications and projects, select the applications or projects in the Explorer view, and then select **File** > **Unregister** from the main workbench menu. **Unregister Application** and **Unregister Project** actions are also available when you right-click the selected items in the Explorer view.

Note: Unregistering an item does not remove any published data from the AppScan Source database.

Publishing assessments to AppScan Source

You can publish assessments to the AppScan Source Database for the purpose of storing and sharing assessments.

About this task

Applications and projects must be registered with AppScan Source before assessments of them can be published. See "Registering applications and projects for publishing to AppScan Source" for more information. By default, if you attempt to publish an assessment of unregistered applications or projects, you will be prompted to register the applications or projects at that time (which requires **Register** permission).

Note: Assessments that are created as a result of scanning individual files cannot be published.

Restriction: When you scan multiple applications or projects, a parent node containing assessments for each scanned item is created in the My Assessments view. The individual child assessments cannot be managed in this case (for example, the child assessments cannot be removed or published individually). When multiple applications or projects are scanned at the same time, you can only manage the assessments as a group (the parent node).

Procedure

1. To publish the assessment that is currently open in the Triage perspective, select **File** > **Publish Assessment to AppScan Source** in the main workbench menu.
2. To publish an assessment in the My Assessments view, select it and click the view's **Publish Assessment to AppScan Source** button - or right-click the assessment and select **Publish Assessment to AppScan Source**.

Results

When saving an assessment, AppScan Source for Analysis writes absolute paths to the assessment file to reference items such as source files. These absolute paths may cause difficulty in sharing the file on another computer that has a different directory structure. To be able to create portable assessment files, you should create a variable (see "Defining variables" on page 68 or "Defining variables when publishing and saving" on page 96).

Once published, the assessment listed in the My Assessments view will have an icon in the **Published** column. In addition, the assessment will appear in the Published Assessments view, which is a filter-driven view of assessments published to the AppScan Source Database. This view can be set to display only the assessments that match the filter criteria. For example, if 1,000 assessments are published, and you only want to view the assessments that you published, you could create a filter with **By Publisher** as the criteria and **Current User** or your user name as the value.

Setting a filter in the Published Assessments view

Filters can be used to limit the number of assessments that are displayed in the Published Assessments view.

Procedure

1. In the Published Assessments view, click the toolbar **Set Filter** button.
2. Select the check box (or check boxes) for the filter criteria that you want:
   - **By Application**: Select the application for which you want assessments to display. Assessments that have been generated for multiple applications appear if the specified application was part of the assessment.
   - **By Publisher**: Set the view to display assessments that have been published by the current user - or specify a user whose published assessments you want to display.
   - **By Date Proximity**: Specify a date range relative to the current date in hours, days, weeks, months, or years. You can select **By Date Proximity** or **By Date Range**, but not both.
   - **By Date Range**: Specify a range of assessment dates to display in the view. You can select **By Date Proximity** or **By Date Range**, but not both.
3. Click **OK** to set the filter.

Results

Click **Refresh Filter** after applying the filter criteria to refresh the view with any added or removed assessments since the filter was last applied. Click **Clear Filter** to remove an existing filter and show all assessments.

Deleting published assessments from AppScan Source

If you have published an assessment to AppScan Source, you can use actions in the Published Assessments view to remove them.

Procedure

1. In the Published Assessments view, select the assessment that you want to delete. Multiple assessments can also be selected using keyboard Ctrl or Shift keys.
2. Select the **Delete Assessments** button in the view's toolbar - or right-click the selection and choose **Delete Assessments** from the menu.

Publishing assessments to the AppScan Enterprise Console

If your AppScan Enterprise Server has been installed with the Enterprise Console option, you can publish assessments to it. The Enterprise Console offers a variety of tools for working with your assessments - such as reporting features, issue management, trend analysis, and dashboards.

About this task

Before you can publish assessments to the Enterprise Console, you must configure server settings in the AppScan Enterprise Console preference page. For information about setting preferences, see “AppScan Enterprise Console preferences” on page 65.

Note: When publishing to the AppScan Enterprise Console, the version and release level of AppScan Source must match that of the AppScan Enterprise Server. For example, you can publish from AppScan Source Version 8.6.x to AppScan Enterprise Server Version 8.6.x - but you cannot publish from AppScan Source Version 8.6.x to AppScan Enterprise Server Version 8.5.x.

Restriction: When you scan multiple applications or projects, a parent node containing assessments for each scanned item is created in the My Assessments view. The individual child assessments cannot be managed in this case (for example, the child assessments cannot be removed or published individually). When multiple applications or projects are scanned at the same time, you can only manage the assessments as a group (the parent node).

Procedure

1. Use one of these methods to publish one or more assessments to the Enterprise Console:
   a. Select one or more assessments in the My Assessments view and then click **Publish Assessment to AppScan Enterprise Console**.
   b. Right-click the assessment (or a selection of assessments) in the My Assessments view and select the **Publish Assessment to AppScan Enterprise Console** menu item.
   c. When an assessment is open, choose **File** > **Publish Assessment to AppScan Enterprise Console** from the main menu.
2. In the Publish to AppScan Enterprise Console dialog box:
   a. Specify an AppScan Enterprise Console application to associate the assessment with. By default, the application is set to the last application that

was specified for publishing. If no applications have previously been specified when publishing, no application will be used by default. To specify an application:

1) Click the **Application** field **Select** button.
2) The Select Application dialog box opens, displaying all applications that already exist in the AppScan Enterprise Console. To view an application's attributes in the AppScan Enterprise Console, click **View Profile** next to it.
3) Select the application to associate the scan with - or create a new application for this purpose by clicking **Create new application**. Clicking this link will open the AppScan Enterprise Console and allow you to create a new application. Once the new application's attributes have been saved, the Select Application dialog box will automatically refresh to include it for selection (if it does not automatically include the new application, click **Refresh**).

   Tip: In the Select Application dialog box, you can use the filter field to narrow down the list of applications. As you type, the filter is automatically applied to the list of applications. The asterisk (*) and question mark (?) characters can be used as wildcards. An asterisk matches any group of zero or more characters, while a question mark matches any single character.
4) Click **OK** after you have selected the application.

b. Use the **Folder** field to set the location to publish to. By default, the location is set to the last location that was used for publishing. If no assessments have previously been published, your default AppScan Enterprise Console folder is selected (note that this is the default folder for the user ID that is specified in the AppScan Enterprise Console preference page). To choose a different folder to publish to, click the **Folder** field **Select** button and then choose the folder that you want (only folders that you have permission to publish to are available). If the folder that you want to publish to is not available, click **Refresh** to update the folder tree with any changes that have been made on the server.

c. In the **Name** field, specify a name that the assessment will be saved as in the AppScan Enterprise Console.

3. Click **Publish**.

Results

When saving an assessment, AppScan Source for Analysis writes absolute paths to the assessment file to reference items such as source files. These absolute paths may cause difficulty in sharing the file on another computer that has a different directory structure. To be able to create portable assessment files, you should create a variable (see “Defining variables” on page 68 or “Defining variables when publishing and saving” on page 96).

After the assessment has been published, a link to AppScan Enterprise (Enterprise Console) will be provided in an information message. Clicking the link will open the portal page in your default external web browser.

Tip: If publishing fails, check that the Enterprise Console server is running and that you are able to access its control center URL in a browser (use the same **Enterprise Console URL** that you have specified in the AppScan Enterprise Console preference page).

Note:

- Large assessments may take longer to appear at the portal. If you receive no error messages after publishing and the report does not appear at the portal, check with your administrator.
- Any attempts to publish an assessment that has the same name as one that is currently being processed by the Enterprise Console will fail. In addition, if you publish the commonly-named assessment after the first one has been processed, the second assessment will overwrite the first one (the Enterprise Console can provide a trending analysis for commonly-named reports if it has been configured to do so ahead-of-time). To determine if an assessment has finished processing, access the Enterprise Console control center in a web browser and then navigate to the appropriate user folder and check the status of the report.
- AppScan Source does not support publishing to an Enterprise Console instance that has been configured to use proxy settings. Attempting to publish to an instance that uses proxy settings will result in an error.

AppScan Enterprise Console preferences

If your AppScan Enterprise Server has been installed with the AppScan Enterprise Console option, you can publish assessments to it. The Enterprise Console offers a variety of tools for working with your assessments - such as reporting features, issue management, trend analysis, and dashboards.

To enable this feature, complete the AppScan Enterprise Console preference page. The **Domain** field in this page is optional. The remaining fields in this page must be completed with valid entries before Enterprise Console publication is enabled:

- **User ID** field: Enter the user name that you use to connect to the Enterprise Console. At a minimum, you must be a QuickScan user with your own user folder on the Enterprise Server.
- **Password** field: Enter the password used to log in to Enterprise Console (the password for the user name that was entered).
- **Domain** field: Enter the domain of the machine on which the Enterprise Console is installed.
- **Enterprise Console URL** field: Enter the URL used to access the Enterprise Console web application.

  The format of this URL is:

  `http(s)://<hostname>:<port>/ase`

  Where `<hostname>` is the name of the machine on which the Enterprise Console has been installed and `<port>` is the port on which the console is running (`<port>` does not need to be specified if the Enterprise Console is running on its default port). An example of this URL is `http://myhost.mydomain.ibm.com/ase`.

  Note:

  - This field is optional if the **Enterprise Console URL** has already been set.
  - You must be signed into AppScan Source with **Manage AppScan Enterprise Settings** permission to be able to set the **Enterprise Console URL** field. For information about user accounts and permissions, see the *Administering* section of the product infocenter - or the *Administering AppScan Source* section of the *IBM Security AppScan Source Installation and Administration Guide*.
  - The **User ID** and **Password** are stored on the machine that is running the AppScan Source client (for example, AppScan Source for Analysis) - while the **Enterprise Console URL** is stored in the Enterprise Server (which may be

located on a remote machine). You cannot access user name and password information from the remote machine (for example, by issuing the `getaseinfo` command from it).

– AppScan Source does not support publishing to an AppScan Enterprise Console instance that has been configured to use proxy settings. Attempting to publish to an instance that uses proxy settings will result in an error.

After completing the settings, it is strongly recommended that you ensure that the connection to the Enterprise Console server is valid by clicking **Test Connection**.

Tip: If the connection test fails, check that the Enterprise Console server is running and that you are able to access its control center URL in a browser (use the same **Enterprise Console URL** that you specified above).

Saving assessments

Before you begin

Important: To save assessments, you must have **Save Assessments** permission. To learn about setting permissions, see the *IBM Security AppScan Source Installation and Administration Guide*.

About this task

Assessments can be saved locally and then opened again at any time. By default, assessments are saved with a `.ozasmt` file extension to your operating system home directory (for example, on Windows, the directory might be `C:\Documents and Settings\Administrator\`).

Procedure

1. To save the assessment that is currently open in the Triage perspective, select **File** > **Save Assessment** or **File** > **Save Assessment As** in the main workbench menu. Choosing the **Save Assessment As** action allows you to specify the location and filename of the saved assessment.
2. To save an assessment in the My Assessments view, select it and click the view's **Save Assessment** or **Save Assessment As** button - or right-click the assessment and select **Save Assessment** or **Save Assessment As**.

Results

When saving an assessment, AppScan Source for Analysis writes absolute paths to the assessment file to reference items such as source files. These absolute paths may cause difficulty in sharing the file on another computer that has a different directory structure. To be able to create portable assessment files, you should create a variable (see “Defining variables” on page 68 or “Defining variables when publishing and saving” on page 96).

Automatically saving assessments

By default, scans are saved automatically to `<data_dir>\scans` (where `<data_dir>` is the location of your AppScan Source program data, as described in “Installation and user data file locations” on page 258) for a period of three days. This behavior is determined by the `assessment_auto_save`, `assessment_auto_save_location`, and `assessment_auto_save_stale_period` settings in `<data_dir>\config\scanner.ozsettings`.

- When the `assessment_auto_save` setting is set to `true`, assessments are automatically saved when they complete (you must have **Save Assessments** permission).
- The `assessment_auto_save_location` setting determines the location in which the assessments are stored. By default, assessments are saved to `<data_dir>\scans`. To change this location, set the `value` attribute to a directory of your choice. For example, to set the location to `C:\myFolder`, set the attribute to `value="C:\myFolder"`.
- The `assessment_auto_save_stale_period` setting determines the number of days that assessments will be kept in the `assessment_auto_save_location`. You can alter this setting with the `value` attribute. For example, saved assessments will be removed from `assessment_auto_save_location` after 10 days if the attribute is set to `value="10"`.

Removing assessments from My Assessments

When assessments are removed from the My Assessments view, they are not removed from your local file system. If an assessment is removed from the view, it can be added back with the **Open Assessment** action.

About this task

Restriction: When you scan multiple applications or projects, a parent node containing assessments for each scanned item is created in the My Assessments view. The individual child assessments cannot be managed in this case (for example, the child assessments cannot be removed or published individually). When multiple applications or projects are scanned at the same time, you can only manage the assessments as a group (the parent node).

Procedure

1. In the My Assessments view, select the assessment that you want to remove. Multiple assessments can also be selected using the keyboard Ctrl or Shift keys.
2. Select the **Remove from My Assessments** button in the view's toolbar - or right-click the selection and choose **Remove from My Assessments** from the menu.

Defining variables

When saving assessments or bundles, or publishing assessments, AppScan Source for Analysis may suggest that you create a variable to replace absolute paths (without variables, AppScan Source for Analysis writes absolute paths to the assessment file to reference items such as source files). When you configure variables for absolute paths, you facilitate the sharing of assessments on multiple computers. It is recommended that you use variables when sharing assessments.

About this task

Variables can be created prior to initiating a save or publish action by following the instructions in this topic - or they can be created after initiating the save or publish action by following the steps in “Defining variables when publishing and saving” on page 96.

To learn, by example, how variables can assist when sharing assessments, see “Example: Defining variables” on page 96.

Procedure

1. Select **Edit** > **Preferences** from the main menu. In the Preferences dialog box, choose **Change Variables**.
2. Click the **Add Variable** button in the Change Variables preference page.
3. Enter a name for the variable and browse to the file location that will be replaced by the variable (AppScan Source for Analysis inserts the surrounding percent symbols (%) after the variable is created).
4. Repeat the above step for any other reference items in the assessment (for example, if the assessment references source in multiple locations, add a variable for each location).
5. Use the preference page to edit and remove variables, using the **Modify Variable** and **Delete Variable** buttons.
6. Click **OK** when you are finished defining variables.

Defining variables when publishing and saving

When you attempt to save or publish an assessment, AppScan Source for Analysis detects any absolute paths in the assessment. If corresponding variables have not been created for the absolute paths, you will be prompted to create them.

About this task

Variables can be created prior to initiating a save or publish action by following the instructions in "Defining variables" on page 68 - or they can be created after initiating the save or publish action by following the steps in this topic.

To learn, by example, how variables can assist when sharing assessments, see "Example: Defining variables."

Procedure

1. After initiating the save or publish action, click **Yes** in the Absolute Paths Detected message.
2. In the Define Variables dialog box, AppScan Source for Analysis suggests a set of paths that encompass the data.
3. Select a directory and click **Add Variable**.
4. Repeat the above step for any other reference items in the assessment (for example, if the assessment references source in multiple locations, add a variable for each location).
5. The Define Variables dialog box can also be used to edit and remove variables, using the **Modify Variable** and **Delete Variable** buttons.
6. Click **OK** to complete the save or publish action.

Example: Defining variables

To share assessment data, you must define the appropriate variables. The example in this topic illustrates the need for a variable.

User Joe scans on computer A, where all source code exists under the directory `C:\dev\my_code`. Joe wants to save his scan results to a file and share it with Bill. Bill uses computer B and has the same code that Joe scanned under directory `C:\code\bills_code`. Without variables, the assessment file will reference all of the source files with absolute paths starting with `C:\dev\my_code`. If Bill opens this assessment file on computer B, AppScan Source for Analysis cannot locate the source files since they exist under `C:\code\bills_code` on computer B.

Solution

Both Joe and Bill should create a variable that points to the root of the source code. Joe creates a variable in AppScan Source for Analysis named `SRC_ROOT`, and gives it a value of `C:\dev\my_code`. This variable is local to Joe's AppScan Source for Analysis installation. Joe then tells Bill the variable name (`SRC_ROOT`) and the location that it points to. Bill then creates a variable named `SRC_ROOT` with a value of `C:\code\bills_code`, in his AppScan Source for Analysis. When Joe saves his scan, the variable `SRC_ROOT` replaces the path `C:\dev\my_code`. When Bill opens the assessment file that he received from Joe, `C:\code\bills_code` substitutes for the `SRC_ROOT` variable.

Chapter 5. Triage and analysis

Grouping similar findings allows security analysts or IT auditors to segment and triage source code problems. This section explains how to triage AppScan Source assessments and analyze results.

When you scan code, the scan results or *findings* appear. *Triage* is the process of evaluating the findings and determining how to resolve them. However, the steps required to reach this goal depend on multiple factors, including the total number of findings, specific security concerns, application risk assessment, and so forth. In addition to deciding whether a finding represents a valid security issue, triage also involves modifying attributes of findings (severity, type, classification) when appropriate.

A triage strategy is important to ensure that you accomplish your goals in the order and time period that you want. Triage is best accomplished in an iterative fashion where you evaluate a subset of findings and determine the disposition of each subset in each iteration. There are many valid approaches for deciding how to define the triage iterations. One approach is to create subsets of the high risk findings based on overall severity. You could start to resolve findings that potentially present the most risk and move to those that present the least likely risk. Another approach is to define subsets by security concern, such as SQL Injection or Validation Required.

Typically, a security analyst or IT auditor performs triage. The analyst or auditor may submit the findings that require code changes to a defect tracking system, and then to developers for remediation. In other instances, developers may triage and resolve issues.

During the triage phase, you can:
- Review findings of particularly interesting vulnerability types
- View APIs in a particular category
- Compare findings in different assessments
- Filter or exclude specific findings
- Change the severity or vulnerability type of a finding
- Promote suspect and scan coverage findings to definitive
- Annotate findings
- Submit defects to defect tracking systems or email findings to others.

AppScan Source provides all of the necessary tools to analyze results using a variety of triage strategies. Filtering provides a means to view only the findings to be processed within a specific triage iteration. If your iterative strategy is by severity and classification, you can filter the findings from the Vulnerability Matrix view. If your iterative strategy is by Vulnerability Type, you can filter from the Assessment Summary view. AppScan Source for Analysis also provides a filter editor to support complex iterative approaches.

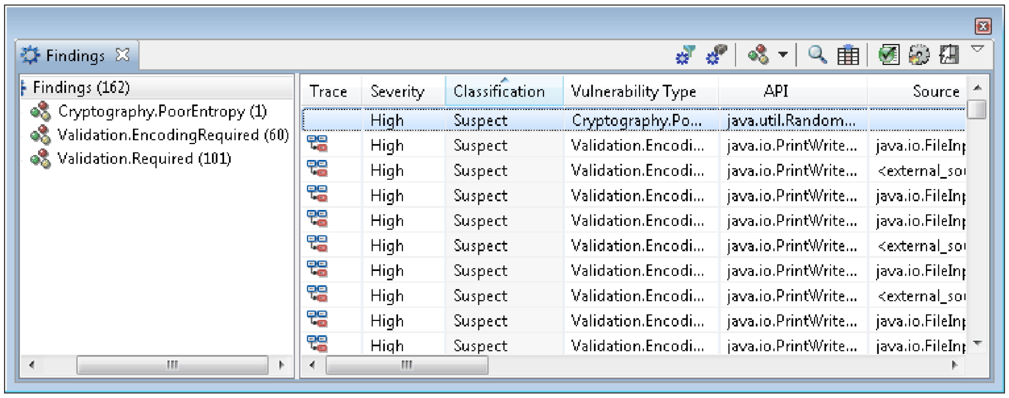
Once you select your triage approach, AppScan Source for Analysis supports the disposition of findings.
- Exclude individual or collections of findings

- Modify the finding details (type, severity, classification)
- Create bundles (a grouping mechanism for findings)
- Compare assessments with the Assessment Diff view

Displaying findings

The Findings view, or any view with findings, displays a findings tree (a hierarchical grouping of assessment criteria) and a findings table for each scan. The item that is selected in the findings tree determines the findings that are presented in the table.

Selecting the root of the tree causes all findings to display in the table - and selecting a grouping type causes only those types of findings to display.


Findings

- Findings (162)
  - Cryptography.PoorEntropy (1)
  - Validation.EncodingRequired (60)
  - Validation.Required (101)

Trace	Severity	Classification	Vulnerability Type	API	Source
	High	Suspect	Cryptography.Po...	java.util.Random...	
▣	High	Suspect	Validation.Encodi...	java.io.PrintWrite...	java.io.FileInp
▣	High	Suspect	Validation.Encodi...	java.io.PrintWrite...	<external_sou
▣	High	Suspect	Validation.Encodi...	java.io.PrintWrite...	java.io.FileInp
▣	High	Suspect	Validation.Encodi...	java.io.PrintWrite...	java.io.FileInp
▣	High	Suspect	Validation.Encodi...	java.io.PrintWrite...	<external_sou
▣	High	Suspect	Validation.Encodi...	java.io.PrintWrite...	java.io.FileInp
▣	High	Suspect	Validation.Encodi...	java.io.PrintWrite...	<external_sou
▣	High	Suspect	Validation.Encodi...	java.io.PrintWrite...	java.io.FileInp
▣	High	Suspect	Validation.Encodi...	java.io.PrintWrite...	java.io.FileInp

AppScan Source for Analysis displays findings by different groupings that include:
- **Vulnerability Type**
- **Classification**
- **File**
- **Source**
- **Sink**
- **API**
- **Bundle**
- **CWE**
- **Table**

Note: Classification and severity sort in descending order by default. All other columns sort in ascending order.

These columns appear in a findings table.

Table 11. Findings table

Column Heading	Description
Trace	An icon in this column indicates that a trace exists for lost or known sinks.

Table 11. Findings table (continued)

Column Heading	Description
Severity	• `High` : Poses a risk to the confidentiality, integrity, or availability of data and/or the integrity or availability of processing resources. High-severity conditions should be prioritized for immediate remediation. • `Medium` : Poses a risk to data security and resource integrity, but the condition is less susceptible to attack. Medium-severity conditions should be reviewed and remedied where possible. • `Low` : Poses minimal risk to data security or resource integrity. • `Info` : The finding, itself, is not susceptible to compromise. Rather, it describes the technologies, architectural characteristics, or security mechanisms used in the code.
Classification	Type of finding: **Definitive** or **Suspect** security finding - or **Scan Coverage** finding. **Note:** In some cases, a classification of **None** may be used to denote a classification that is neither a security finding or a scan coverage finding.
Vulnerability Type	Vulnerability category, such as `Validation.Required` or `Injection.SQL`.
API	The vulnerable call, showing both the API and the arguments passed to it.
Source	A source is an input to the program, such as a file, servlet request, console input, or socket. For most input sources, the data returned is unbounded in terms of content and length. When an input is unchecked, it is considered tainted.
Sink	A sink can be any external format to which data can be written out. Sink examples include databases, files, console output, and sockets. Writing data to a sink without checking it may indicate a serious security vulnerability.
Directory	Full path of the scanned files.
File	Name of the code file in which the security finding or scan coverage finding occurs. File paths in findings are relative to the scanned project working directory.
Calling Method	The function (or method) from which the vulnerable call is made.
Line	Line number in the code file that contains the vulnerable API.
Bundle	Bundle that contains this finding.

Table 11. Findings table (continued)

Column Heading	Description
CWE	ID and topic of the community-developed dictionary of common software weaknesses (Common Weakness Enumeration (CWE) topics).

The AppScan Source triage process

The triage process includes manipulating findings through bundles, filters, and exclusions - and comparing assessment results.

Filters

A *filter* is a set of rules that defines findings with certain traits. A filter allows you to present a dynamic view of these findings and allows you to triage similar findings.

Filters are either *shared* or *local*:

- Shared filters reside on the AppScan server. Anyone connected to that server may use the filter.
- Local filters reside on the local computer.

Bundles

A *bundle* is a named collection of individual findings that is stored with an application. A bundle is created by simply selecting findings and adding them to a new or existing bundle.

Grouping similar findings into bundles allows security analysts to segment and triage source code problems. You can submit bundles to a defect tracking system or email the findings to developers for review as part of the triage and analysis process.

Exclusions

An *exclusion* omits findings from scans. AppScan Source has a built-in **Excluded Bundle**, which contains any findings that you exclude (for example, because they do not require resolution).

Note: Findings excluded from assessment results do not contribute to the calculation of application or project metrics.

Modified findings

A *modified* finding is a finding with an altered vulnerability type, severity, or classification. If you add notes to findings, the finding is also considered modified.

Comparing assessments

Assessments are compared in AppScan Source for Analysis using the **Diff Assessments** action. When two assessments are compared, the differences between the two are displayed in the Assessment Diff view (which resembles a combination of the My Assessments view and the Findings view).

Note: When assessments are compared, filters and bundles are ignored.

Sample triage

This example describes an AppScan Source triage workflow used by a security analyst. Triage workflow may vary according to your business needs.

Mr. Jones, the company security analyst, wants to triage his scan results. He wants to group and prioritize similar findings and then submit them to the appropriate developer for resolution.

First, Mr. Jones scans the source code for his application and then opens the assessment in the Triage perspective. The scan generates about 2,000 findings, all of which he can review in the Findings view. However, Mr. Jones wants first to get an overview of the results and opens the Vulnerability Matrix view that shows the breakdown by severity and finding type (security or scan coverage). Scan coverage findings and suspect security findings require further investigation to determine risk.

In the Vulnerability Matrix, Mr. Jones sees eight high-severity definitive security findings. He clicks the matrix box that indicates eight definitive findings, automatically creating a filter and causing the Findings view to refresh and display only those eight critical issues. Mr. Jones decides to treat these problems as bugs. He selects all eight and submits them to his defect tracking system. He then resets his filter from the Vulnerability Matrix.

Mr. Jones then focuses on the Assessment Summary view. He notices that the 2,000 findings consist of more than half a dozen vulnerability types. He decides to concentrate on validation issues and creates another filter from the Assessment Summary view. He clicks `Validation.EncodingRequired` and `Validation.Required` on the graph and reduces the number of findings in the Findings view to about 500 findings.

Five hundred findings are still difficult to triage. Mr. Jones decides to filter the results further. In the Filter Editor view, he augments the filter created from the Assessment Summary with the requirement for high severity. The findings table now displays 150 entries.

When he sorts by file name, he notices that some of the findings are in code from a third party library. Mr. Jones knows that the use of this library is isolated and that he does not intend to address its security issues. He excludes these findings, causing the Findings view and metrics to update immediately. Future scans will detect these findings, but they are segregated and will not contribute to metrics.

Mr. Jones notices several high severity suspect security findings of type `Validation.Required`. He knows that data is being consumed without validation. He decides to promote these findings from suspect to definitive. While making this modification, he decides to add notes to explain his changes, and then he emails these findings to himself as a reminder to prioritize their remediation or review them in the Modified Findings view.

Next, Mr. Jones sorts by file name again and notices that some of the findings are in the backend server and some are in the user interface. He selects all backend findings, and creates a new bundle, labeled `Backend Server - Validation Required`. He selects the remaining findings and places them in a bundle labeled `UI`

- Validation Required. Triage continues with a focus on Validation.EncodingRequired types with high severity.

At the end of the day, Mr. Jones has created a dozen bundles. Throughout the day, he uses the graph, filters, and Vulnerability Matrix to prune the findings to a manageable number in view at one time. Sometimes he places these individual findings in bundles. Other times he excludes unimportant findings. At times, he creates new bundles for specific findings; sometimes he adds findings to an existing bundle.

Now Mr. Jones reviews the dozen bundles. He determines that he should submit the Backend Server - Validation Required and UI - Validation Required bundles to his defect tracking system to notify the developers of these areas of concern.

Mr. Jones goes to the Bundles view and opens the Backend Server - Validation Required bundle. A new view entitled **Backend Server - Validation Required** opens with a list of the findings that he placed in the bundle. He then submits this bundle to a defect tracking system. Later that night, when the developer logs in to Rational ClearQuest and sees the bugs assigned to him, he can open the finding in AppScan Source for Development.

Mr. Jones reviews the other bundles. He submits some to the defect tracking system and emails others to his colleagues. However, some bundles contain findings that upon further review are not that important to him. He moves these less important findings into two new bundles, By Design and Irrelevant. Mr. Jones determined that these findings are acceptable, and he does not intend to alter the code. In addition to the By Design and Irrelevant findings, Mr. Jones realizes that all Cryptography.PoorEntropy findings are unimportant to him too. He knows that the entropy may be poor for those cryptography calls, and although a fast computer could crack the key in less than a week, it is not important for an application in which the data is no longer useful a few hours after encrypting it. Mr. Jones wants to remove these too.

He then adds the By Design and Irrelevant bundles to the **Excluded Bundles** list in the Properties view. He also opens the Filter Editor and creates another filter with the vulnerability type, Cryptography.PoorEntropy, saves the filter named Crypto, and sets the behavior of the Crypto filter to **Inverted** (in the Select Filter dialog box, he chooses **Invert filter**). He then starts a scan and goes home. The metrics do not reflect these exclusions until after the next scan.

Triage with filters

AppScan Source for Analysis reports on all potential security vulnerabilities and may produce many thousands of findings for a medium to large code base. When you scan, you may find that the findings list contains items that are not important to you. To remove certain findings from the Findings view, you can choose a predefined filter or you can create your own *filter*. A filter specifies the criteria that determine which findings to remove from view.

- “Filter overview” on page 105
- “Filter rules” on page 105
- “Filter examples” on page 107

Filter overview

Filters help you manage scan results during triage or reporting. A filter helps guide workflow and focuses security analysts on the most critical areas of a subset of findings. For example, during code examination, an analyst may create a filter to avoid viewing low severity findings. Alternatively, the analyst may prefer to exclude vulnerabilities in system library `include` files. A filter can eliminate these items from view, and may exclude individual files or files that have been previously investigated.

A filter removes or restricts items that meet the criteria determined by the filter rules. All filtered items remain in the scan results - but only appear in the Findings view if the **Show Filtered Findings** ( ) toggle is selected.

AppScan Source includes several predefined filters that can be selected in the Filter Editor view.

Once you have a filter, you can set a property to make the filter an *exclusion*. An exclusion affects scans and eliminates all findings that match the filter or alternatively, all findings that do not match the filter.

Filter rules

Each filter consists of rules that define which findings to restrict (include) or remove (exclude) from the results in the findings table (for trace rules, you can both restrict and remove based on trace properties).

- A **Restrict to** rule (inclusive rule) excludes findings that do not have the specified criteria and removes these findings from the visible results in the findings table.
- A **Remove** rule (exclusive rule) removes findings that contain criteria from the scan results. A remove rule excludes any findings that have the specified criteria and removes these findings from the visible results.

A filter rule may include these traits:

- **Severity**: Identifies the potential impact or risk of the individual findings. Severity rules are restrict-only.
  - `High` : Poses a risk to the confidentiality, integrity, or availability of data and/or the integrity or availability of processing resources. High-severity conditions should be prioritized for immediate remediation.
  - `Medium` : Poses a risk to data security and resource integrity, but the condition is less susceptible to attack. Medium-severity conditions should be reviewed and remedied where possible.
  - `Low` : Poses minimal risk to data security or resource integrity.
  - `Info` : The finding, itself, is not susceptible to compromise. Rather, it describes the technologies, architectural characteristics, or security mechanisms used in the code.
- **Classification:** Filter the findings based on classifications described in this topic. Classification rules are restrict-only.
- **Vulnerability Type**: Filter by a particular vulnerability category, such as `BufferOverflow`. When you add a vulnerability type, you can select from all possible vulnerability types - or you can choose from only those types that have been found in the current assessment. To choose from vulnerability types that have been found in the current assessment, select **Show only values from open assessment** in the Select Values dialog box.

Selecting from all possible vulnerability types is useful when you are creating a filter for future scans. To display all vulnerability types, deselect **Show only values from open assessment** (if there are no open assessments, all vulnerability types display by default and the **Show only values from open assessment** check box is unavailable).

- **API**: Filter all vulnerabilities of a specific API.
- **File**: Filter all vulnerabilities from a specific file.
- **Directory**: Filter all vulnerabilities from a specific directory.
- **Project**: Filter all vulnerabilities from a specific project.
- **Trace**: Allows you to filter findings based on trace properties (see "Sources and sinks" on page 141 to learn more about trace properties). Filters can include trace rules that both restrict and remove based on trace properties. When you click **Add** in either section (restrict or remove), the Trace Rule Entry dialog box opens. In it, you can specify:
  - **Source**: In the Source section **API RegEx** field, specify the trace source or a regular expression that covers multiple sources (the default entry is `.*` - the regular expression or wildcard which will return all). If you are using a regular expression, select the type in the **RegEx Type** field menu (the default regular expression type is **PERL**). If you are not using a regular expression, select **Exact Match** in the **RegEx Type** field menu.

    If the **API RegEx** entry is a valid expression, a green check mark icon will appear next to the field. If the entry is not a valid expression, a red X icon will appear next to the field and the dialog box **OK** button will be disabled. Hovering over either icon provides more information about the validation results. If you have made an entry that is not a valid expression, but you want to proceed with its use, select the **Ignore validation errors above** check box at the bottom of the dialog box. This will enable the dialog box **OK** button (so long as the expression is not empty) and the icon next to the invalid expression will change to a green check mark with **Validation disabled** hover text.

    You can also refine the filter by mechanism or technology by using the **Add a VMAT property** button in the Source Properties section (more information about VMAT properties is available below) - however using this feature to limit by vulnerability will not have the desired effect since vulnerability type is determined by sink and not the source.
  - **Sink**: In the Sink section, you can add sinks as filters in the same manner in which sources are specified.

    You can refine the filter by limiting it to specific vulnerability types (to limit the effect of the trace rule entry to just specific types of vulnerabilities, mechanisms, or technologies). To do this, click the **Add a VMAT property** button in the Sink Properties section and then select the property in the Choose Properties dialog box. The list of properties can be filtered using the **Filter** field.

    VMAT is a categorization of the four major types of properties that AppScan Source applies to application programming interfaces (API). VMAT property categories include:
    - **Vulnerability**: the type of exploit or attack vector that results in a security violation
    - **Mechanism**: the security control used to prevent a vulnerability
    - **Attribute**: these properties are not currently available in the Choose Properties dialog box

  - **Technology**: the general description for the type of functionality an API provides

  Filter example: To filter for all SQL injections and XSS coming from HTTP (your highest risk source), create a **Restrict to** trace rule that contains a `Technology.Communications.HTTP` filter in the Source Properties section and `Vulnerability.Injection.SQL` and `Vulnerability.CrossSiteScripting` rules in the Sink Properties section.
- **Required Calls**: In the Required Calls section, add specific API calls that must be on the path from the source to the sink. Required calls limit the findings to those that have traces that pass through the specified required calls. When you click **Add an intermediate call**, the Configure API dialog box opens. In this dialog box, specify the call in the same manner that sources and sinks are specified.
- **Prohibited Calls**: In the Prohibited Calls section, add specific API calls that must not be on the path from the source to the sink. Prohibited calls limit the findings to those that have traces that do not pass through the specified prohibited calls. Add a prohibited call in the same manner as required calls are added.

Tip:
- When filtering by **Vulnerability Type**, **API**, **File**, **Directory**, or **Project**, the list that displays in the Select Values dialog box can be filtered by typing a pattern into the filter field at the top of the dialog box.
- In any findings table, look at the **Source** and **Sink** columns to get a feel for the sources and sinks that you want to filter out.
- To get a feel for source, sink, and call properties that you want to filter, look at the **Vulnerability Type** column in any findings table.
- To see calls that you may want to filter, view the entries in the **API** column in any findings table.

Filter examples

Table 12. Filter examples

Filter behavior in a findings table	Filter settings in the Filter Editor view
Findings table includes only high-severity suspect security findings.	• In the Severity section, select the **High** check box and deselect all other check boxes. • In the Classification section, select the **Suspect** check box and deselect all other check boxes.
Findings table includes all findings in a project named `ProjectA`, with the exception of info vulnerability types.	• In the Vulnerability Type section, select the **Remove** radio button and click **Add**. In the Select Values dialog box, select **Vulnerability.Info**. • In the Project section, select the **Restrict to** radio button and click **Add**. In the Select Values dialog box, select `ProjectA`.

Table 12. Filter examples (continued)

Filter behavior in a findings table	Filter settings in the Filter Editor view
Only findings with a trace are displayed.	In the Trace section, click **Add** in the **Restrict to** section. Accept the default entries in the Trace Rule Entry dialog box and then click **OK**. The default values in the dialog box are: • The Source **API RegEx** field is `.*` and the regular expression type is **PERL**. This tells AppScan Source to filter for any findings with a source (using Perl regular expression syntax). • The Sink **API RegEx** field is `.*` and the regular expression type is **PERL**. This tells AppScan Source to filter for any findings with a sink (using Perl regular expression syntax).
The findings table displays HTTP-related sources to SQL Injection-related sinks that do not pass through `java.lang.Integer.parseInt`.	In the Trace section, click **Add** in the **Restrict to** section. In the Trace Rule Entry dialog box, complete these steps: • In the Source section, click **Add a VMAT property**. In the Choose Properties dialog box, select `Technology.Communications.HTTP`. Click **OK** to add the VMAT property and return to the Trace Rule Entry dialog box. • In the Sink section, click **Add a VMAT property**. In the Choose Properties dialog box, select `Vulnerability.Injection.SQL`. Click **OK** to add the VMAT property and return to the Trace Rule Entry dialog box. • In the Prohibited Calls section, click **Add an intermediate call**. In the Configure API dialog box, enter `java.lang.Integer.parseInt.*` in the **API RegEx** field. Click **OK** to add the intermediate call and return to the Trace Rule Entry dialog box and then click **OK** to add the trace rule entry.

Using AppScan Source predefined filters

AppScan Source includes a set of predefined filters that can be selected from the filter selection list at the top of the Filter Editor view. This help topic describes these out-of-the-box filters.

Note: In AppScan Source Version 8.8, predefined filters were improved to provide better scan results. If you need to continue using the predefined filters from older versions of AppScan Source (archived filters are listed in “AppScan Source predefined filters (Version 8.7.x and earlier)” on page 111), follow the instructions in “Restoring archived predefined filters” on page 113.

Note: In AppScan Source for Development (Visual Studio plug-in), this view is part of the Edit Filters window.

- “CWE SANS Top 25 2010 Vulnerabilities” on page 109
- “External Communications” on page 109

- “Low Severity And Informational”
- “OWASP Mobile Top 10 Vulnerabilities”
- “OWASP Top 10 2010 Vulnerabilities”
- “OWASP Top 10 2013 Vulnerabilities”
- “PCI Data Security Standard V2.0 Vulnerabilities” on page 110
- “Targeted Vulnerabilities - EncodingRequired For HTTP Sources” on page 110
- “! - Important Types” on page 110
- “! - AppScan Vital Few” on page 110
- “! - High Risk Sources” on page 111
- “Targeted Vulnerabilities - Validation Required For C/C++ Sinks” on page 111
- “Trusted Sources” on page 111

CWE SANS Top 25 2010 Vulnerabilities

This filter focuses on vulnerability types related to the *CWE/SANS TOP 25 Most Dangerous Software Errors* for 2010.

To learn about the *2011 CWE/SANS Top 25 Most Dangerous Software Errors*, see http://cwe.mitre.org/top25/.

External Communications

This filter matches findings which originate from outside the application and across a network. This filter matches findings which originate at any `Technology.Communications` source.

Low Severity And Informational

This filter contains findings with severities of Low and Informational. All classifications (Definitive, Suspect, and Scan Coverage) are included.

OWASP Mobile Top 10 Vulnerabilities

This filter focuses on vulnerability types related to the Open Web Application Security Project (OWASP) Mobile Top 10 Release Candidate v1.0 list.

To learn about OWASP, see https://www.owasp.org/index.php/Main_Page. Links to various OWASP documents and security risks are available at https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project.

OWASP Top 10 2010 Vulnerabilities

This filter focuses on vulnerability types related to the Open Web Application Security Project (OWASP) Top 10 2010 list.

To learn about OWASP, see https://www.owasp.org/index.php/Main_Page. Links to various OWASP documents and security risks are available at https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project.

OWASP Top 10 2013 Vulnerabilities

This filter focuses on vulnerability types related to the Open Web Application Security Project (OWASP) Top 10 2013 list.

To learn about OWASP, see https://www.owasp.org/index.php/Main_Page. Links to various OWASP documents and security risks are available at https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project.

PCI Data Security Standard V2.0 Vulnerabilities

This filter focuses on vulnerability types related to the Payment Card Industry Data Security Standard (PCI DSS) Version 2.0 standard.

See https://www.pcisecuritystandards.org/security_standards/index.php for information.

Targeted Vulnerabilities - `EncodingRequired` For HTTP Sources

This filter focuses on findings from the `Validation.EncodingRequired` and `Validation.EncodingRequired.Struts` vulnerability categories. Only findings that originate from a `Technology.Communications.HTTP` source are included. The findings are limited to High and Medium severities with Definitive or Suspect classifications.

! - Important Types

This filter contains findings from a broader range of important vulnerability categories. The findings are limited to High and Medium severities with Definitive or Suspect classifications. The specific categories which are included in this filter are:

```
Vulnerability.AppDOS
Vulnerability.Authentication.Credentials.Unprotected
Vulnerability.BufferOverflow
Vulnerability.BufferOverflow.FormatString
Vulnerability.BufferOverflow.ArrayIndexOutOfBounds
Vulnerability.BufferOverflow.BufferSizeOutOfBounds
Vulnerability.BufferOverflow.IntegerOverflow
Vulnerability.BufferOverflow.Internal
Vulnerability.CrossSiteRequestForgery
Vulnerability.CrossSiteScripting
Vulnerability.CrossSiteScripting.Reflected
Vulnerability.CrossSiteScripting.Stored
Vulnerability.FileUpload
Vulnerability.Injection
Vulnerability.Injection.LDAP
Vulnerability.Injection.OS
Vulnerability.Injection.SQL
Vulnerability.Injection.XML
Vulnerability.Injection.XPath
Vulnerability.Malicious.EasterEgg
Vulnerability.Malicious.Trigger
Vulnerability.Malicious.Trojan
Vulnerability.PathTraversal
Vulnerability.Validation.EncodingRequired
Vulnerability.Validation.EncodingRequired.Struts
```

! - AppScan Vital Few

This filter matches findings from some of the most dangerous vulnerability categories. The results are limited to High and Medium severity vulnerabilities. Results with specific sources are removed from the findings. The specific vulnerability categories which are included in this filter are:

```
Vulnerability.CrossSiteScripting
Vulnerability.CrossSiteScripting.Reflected
Vulnerability.CrossSiteScripting.Stored
Vulnerability.Injection.OS
Vulnerability.Injection.LDAP
Vulnerability.Injection.SQL
Vulnerability.Injection.Mail
```

! - High Risk Sources

This filter limits the findings to specific vulnerability types and sources with one of these properties:

```
Technology.Communications.HTTP
Technology.Communications.IP
Technology.Communications.RCP
Technology.Communications.TCP
Technology.Communications.UDP
Technology.Communications.WebService
```

Targeted Vulnerabilities - Validation Required For C/C++ Sinks

This filter focuses on `Validation.Required` vulnerabilities for a set of known C and C++ sinks. The findings are limited to High and Medium severities with Definitive or Suspect classifications.

Trusted Sources

This filter presumes that data coming from certain sources, such as session objects or request attributes, is safe.

AppScan Source predefined filters (Version 8.7.x and earlier)

This topic lists predefined filters that were included in AppScan Source Version 8.7.x and earlier.

If you need to access these filters, follow the instructions in “Restoring archived predefined filters” on page 113.

! - The Vital Few

This filter matches findings from some of the most dangerous vulnerability categories. Only findings which originate in an external network communications source are included. This filter provides a laser-focused starting point for high risk findings. The specific categories which are included in this filter are:

```
Vulnerability.BufferOverflow
Vulnerability.BufferOverflow.FormatString
Vulnerability.PathTraversal
Vulnerability.CrossSiteScripting
Vulnerability.CrossSiteScripting.Reflected
Vulnerability.CrossSiteScripting.Stored
Vulnerability.Injection
Vulnerability.Injection.LDAP
Vulnerability.Injection.SQL
Vulnerability.Injection.OS
Vulnerability.Injection.XML
Vulnerability.Injection.XPath
```

High Priority - External Communications

This filter matches findings which originate from outside the application and across a network. This filter matches findings which originate at any `Technology.Communications` source.

High Priority - Important Types

This filter contains findings from some of the most dangerous vulnerability categories, such as `CrossSiteScripting` and `Injection.SQL`. The specific categories which are included in this filter are:

```
Vulnerability.AppDOS
Vulnerability.Authentication.Credentials.Unprotected
Vulnerability.Authentication.Entity
Vulnerability.BufferOverflow
Vulnerability.BufferOverflow.FormatString
Vulnerability.CrossSiteScripting
Vulnerability.CrossSiteScripting.Reflected
Vulnerability.CrossSiteScripting.Stored
Vulnerability.Injection
Vulnerability.Injection.LDAP
Vulnerability.Injection.OS
Vulnerability.Injection.SQL
Vulnerability.Injection.XML
Vulnerability.Injection.XPath
Vulnerability.PathTraversal
```

Low Priority - Test Code

This filter contains findings from test code. Specific types in this filter include:

```
Vulnerability.Quality.TestCode
```

Noise - Copy-like Operations

This filter contains findings that are concerned with copy-like operations. A copy-like operation occurs when data is taken from a source which may or may not be trusted, but actions performed on the data are trusted.

These patterns are looked for:

```
Technology.Database --> Vulnerability.Injection.SQL
Mechanism.SessionManagement --> Mechanism.SessionManagement
Technology.XML, Technology.XML.DOM, Technology.XML.Schema,
Technology.XML.XPath --> Vulnerability.AppDOS.XML,
Vulnerability.Injection.XML
```

Noise - Logging Issues

This filter contains findings related to error handling. The findings found emanate from an error handling routine to a logging mechanism. This pattern is matched:

```
Mechanism.ErrorHandling -->
Vulnerability.Logging, Vulnerability.Logging.Forge, Vulnerability.Logging.Required
```

Noise - Low Severity

This filter contains findings with a severity of Low. All classifications are included.

Noise - Trusted Source

This filter contains findings that emanate from a trusted source. Only findings that have `java.lang.System.getProperty.*` as their source are included in this filter.

Restoring archived predefined filters

Predefined filters that were provided in AppScan Source prior to Version 8.8 can be added back to the product by following the steps in this task. Once restored on a single machine, they can be managed in the same manner as filters that you create (for example, they can be shared to multiple clients).

About this task

Archived predefined filters are located in `<data_dir>\archive\filters` (where `<data_dir>` is the location of your AppScan Source program data, as described in "Installation and user data file locations" on page 258).

Procedure

1. In `<data_dir>\archive\filters`, locate the filter or filters that you want to restore (AppScan Source filters have a `.off` file extension).
2. Copy the filter or filters to `<data_dir>\scanner_filters`.
3. Restart AppScan Source.

What to do next

To learn how to manage filters (including archived filters that you have restored), see "Creating and managing filters in the Filter Editor view" on page 114.

Creating and managing filters

AppScan Source offers multiple methods for creating filters. The main view for filter creation, the Filter Editor view, provides a robust set of rules which can be manually set and then saved to a filter. The Filter Editor view also provides a mechanism for managing filters that you have created - allowing you to easily modify or remove them. Alternately, you can filter the findings table using views that offer graphical representations of the findings - and then save those filters in the Filter Editor view. When you create a filter, the other views update to reflect the filter properties.

- "Creating, managing, and applying filters in the Filter Editor view"
- "Filtering from the Assessment Summary and Vulnerability Matrix views" on page 114
- "Creating filters in the Sources and Sinks view" on page 114

Creating, managing, and applying filters in the Filter Editor view

The Filter Editor view allows you to create filters by specifying filter rules. Filters created in the Filter Editor view can be saved, modified, and removed. Once a filter is created in this view, it can be applied via drop down menu in the view. See "Creating and managing filters in the Filter Editor view" on page 114.

In AppScan Source for Analysis, you can share filters that you have created to the AppScan Enterprise Server - and access filters that others have shared. In AppScan Source for Development, you can access shared filters if you are running in server mode.

Note: In AppScan Source for Development (Visual Studio plug-in), this view is part of the Edit Filters window.

Filtering from the Assessment Summary and Vulnerability Matrix views

Note:

- The Assessment Summary view is not available on OS X.
- In AppScan Source for Development (Visual Studio plug-in), these views are part of the Edit Filters window.

The Assessment Summary and Vulnerability Matrix views offer a graphical representation of findings. In these views, findings are grouped in different ways. These groups can be selected to filter the findings table so that it displays only those findings that are within the selected group or groups. Any filtering that you do by this method is automatically reflected in the Filter Editor view, from which you can then save the filter settings.

Creating filters in the Sources and Sinks view

Note: The Sources and Sinks view is not available in AppScan Source for Development (Visual Studio plug-in).

The Sources and Sinks view provides the ability to view and filter findings based on a trace of input and output. Filtering that is done in this view can be saved directly in the view. While creating the filter, you have the option of applying it immediately to the scan results.

Creating and managing filters in the Filter Editor view

In this view, you can create, edit, save, delete, and manage filters. If you are using AppScan Source for Analysis, you can share filters and access filters that have been shared by others. In AppScan Source for Development, you can access shared filters if you are using server mode and logged in to the AppScan Enterprise Server.

Procedure

1. In the "Filter Editor view" on page 255 toolbar, click **New**. The new filter name is `Untitled<-number>` (where the first new untitled filter is `Untitled` and the next new untitled filter is `Untitled-1`, and so on).

   Note: In AppScan Source for Development (Visual Studio plug-in), this view is part of the Edit Filters window.
2. Expand the categories and select the criteria that you want for the filter.
3. Click **Save** or **Save As**.
4. Name the filter and click **OK**. The new filter name replaces `Untitled<-number>` in the list of filters.

What to do next

To apply the filter, select it in the Filter Editor view drop down menu.

Note: Filters that are applied outside of the Vulnerability Matrix view may not affect the Vulnerability Matrix view. The Vulnerability Matrix view **Show the counts of filtered findings** toolbar button must be selected for the filter to be reflected in the Vulnerability Matrix view.

Filters can be managed directly in the Filter Editor view by selecting the filter in the list and then working with it - or you can click **Manage Filters** to open the Manage Filters dialog box, which provides a list of saved filters.

- **Modifying filters**: Select the filter in the Filter Editor view or in the Manage Filters dialog box and then modify its filter rules and save the changes.

  Note: Built-in filters cannot be modified or deleted.
- **Deleting filters**: Select the filter in the Filter Editor view or in the Manage Filters dialog box and then click **Delete**. In the Manage Filters dialog box, you can select multiple filters and click **Delete** to remove them at the same time.
- **Creating a filter from another filter**: You can modify a filter and then click **Save As** to save it as a filter with a new name. This allows you to create a new filter by building on the settings of an existing filter. You can do this in both the Filter Editor view and the Manage Filters dialog box.

  Tip: The same thing can be accomplished by opening a filter and using the **Save As** action to save it with a new name. You can then open the new filter and modify it. By choosing this method, you can create a new filter from one of the built-in filters.
- **Reverting filter settings**: If you modify the properties of a filter and want to undo those changes, click **Revert** to return the filter to its last saved settings. This action can be performed in both the Filter Editor view and the Manage Filters dialog box. In the dialog box, if you have multiple filters with unsaved changes, clicking **Revert** will cause all selected filters with unsaved changes to be reverted back to their saved settings.
- **Sharing filters** (AppScan Source for Analysis only): To create a shared filter, open a filter in the Filter Editor and click **Share Filter** on the Filter Editor view toolbar.

  Note: To modify, delete, or create shared filters, you must have must have **Manage Shared Filters** permission. To learn about setting permissions, see the *IBM Security AppScan Source Installation and Administration Guide*.

Filtering from the Assessment Summary view

When a scan completes, you can look at its findings in the Assessment Summary view (this view opens by default in the Triage perspective). In this view, you can create a filter from the bar chart.

About this task

After a scan completes, the "Assessment Summary view" on page 254 contains a graphical bar chart representation of findings. The view can be refined to display findings by vulnerability type, API, project, or file. When you select grouped findings in the Assessment Summary view, the findings table changes to display only those findings that have been selected in the Assessment Summary view.

Note: Filters that are applied outside of the Vulnerability Matrix view may not affect the Vulnerability Matrix view. The Vulnerability Matrix view **Show the counts of filtered findings** toolbar button must be selected for the filter to be reflected in the Vulnerability Matrix view.

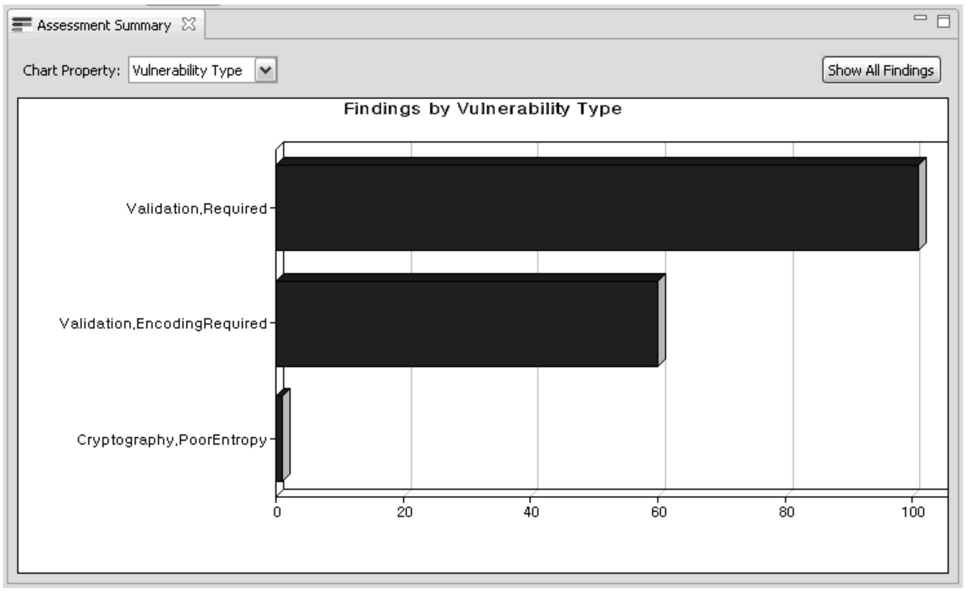
Note:

- The Assessment Summary view is not available on OS X.

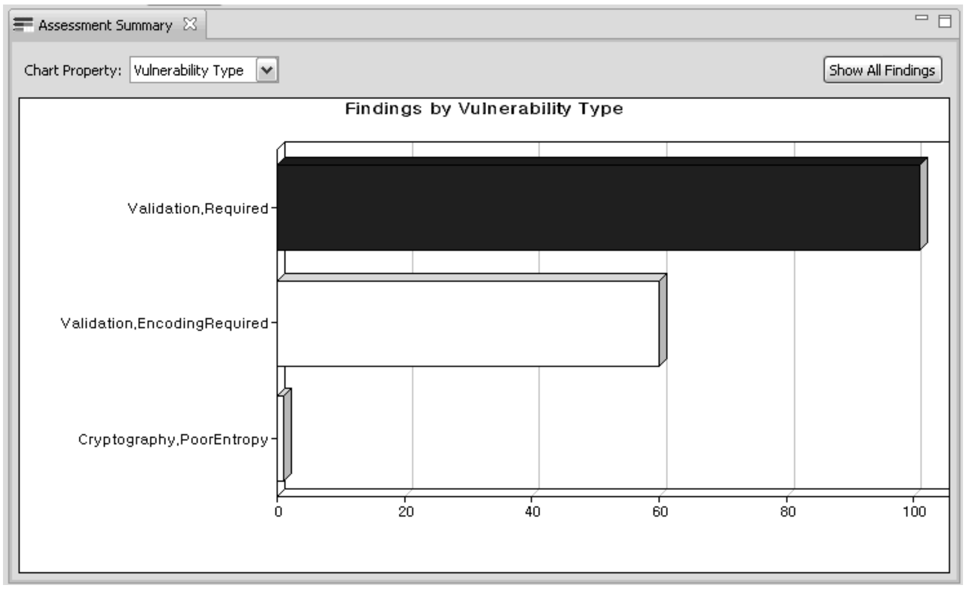
- In AppScan Source for Development (Visual Studio plug-in), this view is part of the Edit Filters window.

Procedure

1. In the Assessment Summary view, change the graphical representation to suit your needs. For example, given an assessment that includes `Validation.Required`, `Validation.EncodingRequired`, and `Cryptography.PoorEntropy` vulnerability types, set the **Chart Property** to **Vulnerability Type**. This will display the findings by vulnerability type in a bar chart representation:

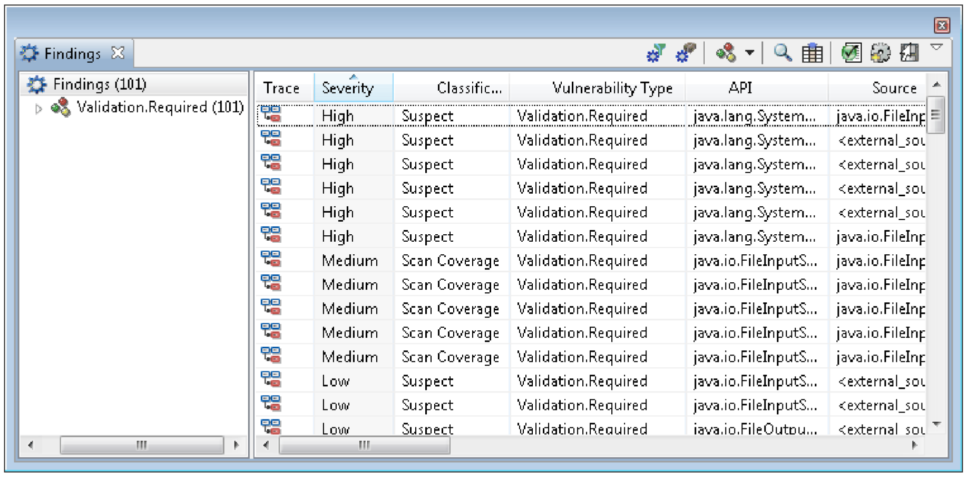


2. To create a filter of `Validation.Required` vulnerability types, click the `Validation.Required` bar in the chart.



Tip: Hold the mouse over the bar to see the number of vulnerabilities.

The filtered results appear in the findings table:

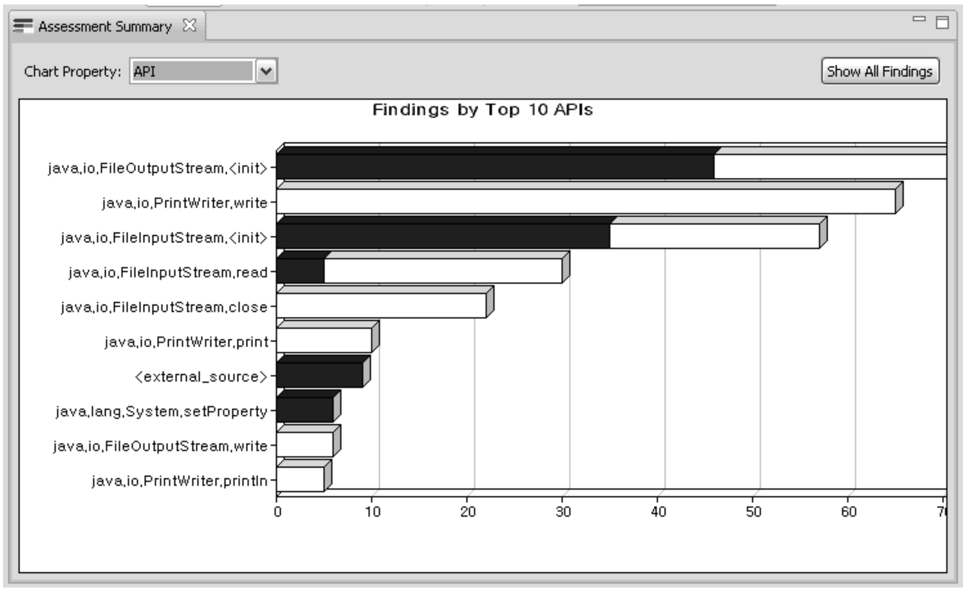


Findings

Findings (101)
Validation.Required (101)

Trace	Severity	Classific...	Vulnerability Type	API	Source
	High	Suspect	Validation.Required	java.lang.System...	java.io.FileInp
	High	Suspect	Validation.Required	java.lang.System...	<external_sou
	High	Suspect	Validation.Required	java.lang.System...	<external_sou
	High	Suspect	Validation.Required	java.lang.System...	<external_sou
	High	Suspect	Validation.Required	java.lang.System...	<external_sou
	High	Suspect	Validation.Required	java.lang.System...	java.io.FileInp
	Medium	Scan Coverage	Validation.Required	java.io.FileInputS...	java.io.FileInp
	Medium	Scan Coverage	Validation.Required	java.io.FileInputS...	java.io.FileInp
	Medium	Scan Coverage	Validation.Required	java.io.FileInputS...	java.io.FileInp
	Medium	Scan Coverage	Validation.Required	java.io.FileInputS...	java.io.FileInp
	Medium	Scan Coverage	Validation.Required	java.io.FileInputS...	java.io.FileInp
	Low	Suspect	Validation.Required	java.io.FileInputS...	<external_sou
	Low	Suspect	Validation.Required	java.io.FileInputS...	<external_sou
	Low	Suspect	Validation.Required	java.io.FileOutpu...	<external_sou

3. The filtering actions also cause the Filter Editor view to populate with the filter rule settings of the selection that was made in the Assessment Summary view. This filter can be saved in the Filter Editor view (to learn about filter rule settings and saving filters, see “Creating and managing filters in the Filter Editor view” on page 114).
4. To view the same filter results by API, set the **Chart Property** to **API**:



Filtering from the Vulnerability Matrix

The Vulnerability Matrix view displays the aggregate number of findings for all applications included in the scan. These findings are grouped in the matrix by severity level. You can create filters by selecting these groups of findings.

About this task

When you select grouped findings in the “Vulnerability Matrix view” on page 255, the findings table changes to display only those findings that have been selected in the Vulnerability Matrix.

Note: In AppScan Source for Development (Visual Studio plug-in), this view is part of the Edit Filters window.

Procedure

1. In the Vulnerability Matrix view, select the section of the matrix that you want to see in the findings table. For example, to see only **High** severity **Suspect** security findings in the findings table, select that section of the matrix. This causes the filtered results to appear in the findings table.
2. The filtering actions also cause the Filter Editor view to populate with the filter rule settings of the selection that was made in the Vulnerability Matrix. This filter can be saved in the Filter Editor view (to learn about filter rule settings and saving filters, see "Creating and managing filters in the Filter Editor view" on page 114).

Creating filters in the Sources and Sinks view

Procedure

1. Open or navigate to the Sources and Sinks view.
2. The Sources and Sinks view contains three sections. The findings table section of the view displays the findings for the sources, sinks, and intermediate nodes that you have chosen to display in the other two sections. This is described in "Sources and Sinks view" on page 249.
3. After you have set the findings table to display the findings that you are interested in, click **Create a new filter based on the selected source, sink, and intermediate nodes**.
4. In the Create Filter dialog box:
   - Specify a name for the filter in the **Name** field.
   - Select **Apply this filter immediately** to have this filter applied to all findings tables in your assessment. Selecting this check box is equivalent to selecting a filter in the Filter Editor view. It sets the current main filter, affecting all views (for example, the Vulnerability Matrix and Findings views).

     Note: Filters that are applied outside of the Vulnerability Matrix view may not affect the Vulnerability Matrix view. The Vulnerability Matrix view **Show the counts of filtered findings** toolbar button must be selected for the filter to be reflected in the Vulnerability Matrix view.
   - If the findings that have been filtered out are irrelevant to your current work, you can remove them from your assessment by selecting the **Create an application filter that excludes these findings** check box. Selecting this check box adds the new filter as an excluded filter in the application properties (in the Properties view for the application, select the Exclusions tab to see the list of excluded filters). For future scans of the application, findings that match the filter will be reported in the Excluded Findings view and not in the Findings view.
5. Click **OK** to filter or exclude the findings.

Applying filters globally

Filters that have already been created can be applied to all applications, individual applications, and to individual projects. Global filters are applied in the Properties view - where you can specify how you want the filters applied (a filter can be applied directly - or its inverse can be applied). For example, if you want to set a global filter for an application, select it in the Explorer view and then open its Properties view (using the **View** menu or by right-clicking the application and clicking **Properties**).

Before you begin

If you are setting a filter for all applications or an individual project, use the Filters tab in the Properties view. If you are setting a filter for an individual application, use the Exclusions and Filters tab in the Properties view.

Procedure

1. In the Filters section of the tab, click **Add**.
2. In the Select Filter dialog box, choose the filter that you want to apply globally.
3. Optional: If you want the inverse of the filter to be applied (rather than applying the filter directly), select **Invert filter**.
4. Click **OK** to close the Select Filter dialog box.
5. When you have completed adding filters, save the changes in the Properties view.

Determining applied filters

Filters can be applied globally to applications and projects before scanning- or they can be applied to assessments after scanning. To allow you to quickly determine how filters have been applied to the findings in an assessment, AppScan Source provides a filter indicator at the bottom of the main workbench.

If no filters have been applied, the filter indicator at the bottom of the workbench indicates that **Findings are not filtered**.

If filters have been applied, the indicator changes to a link that reads **Findings are filtered**. Selecting this link opens a message that allows you to determine how filters have been applied:

- **Scan-time filters** are global filters that are applied to applications and projects:
  - If the assessment is the result of scanning an application or project that does not have filters configured, the message indicates that no scan-time filters have been applied.
  - If the application or project that was scanned does have filters configured, the configured filters are listed by name.
  - In some cases, AppScan Source detects that scan-time filters have been applied, however, the assessment contains no information about those filters. For example, this could happen when an old assessment is opened.
- **Current filters** are filters that have been applied to the findings after scanning. The message indicates if no current filters have been applied - or if filters have been applied. If the latter, a **reset** link is available. When this link is selected, current filters are removed from the findings.

Triage with exclusions

After a scan, you may decide that some findings are irrelevant to your current work, and you do not want them visible in the findings table when you triage the scan results. These *exclusions* (or excluded findings) no longer appear in the Findings view and the assessment metrics update immediately with the changed results. Filter and bundle exclusions added to a configuration only take effect on subsequent scans.

The scope of exclusions

Exclusions apply to all applications (global), individual applications, or projects.

- **Global exclusions** apply to all scans.
- **Application exclusions** only apply to a scan run against a specific application and its corresponding projects.
- **Project exclusions** apply to findings that exist in a specific project.

Note: Exclusions affect assessment metrics, including total findings (findings that are excluded are not included in assessment metrics).

Global Exclusions

You can store or access a global exclusion from any AppScan Source for Analysis application - and these exclusions apply to all scans. Only shared filters may become global filters.

Application and Project Exclusions

Bundle exclusions apply only to an application. A filter exclusion may apply to an application or project. Exclusions that are applied to applications and projects can be shared or local.

Specifying exclusions

Findings can be marked as exclusions from a findings table or the Properties view. Exclusions may consist of individual findings, filters, or bundles. Typically, exclusions created from a findings table take effect immediately. Exclusions created in the Properties view require an additional scan to take effect.

An exclusion applies to the application immediately during these procedures:

- Select one or more findings, right-click the selection, and then choose **Exclude Findings** from the menu.
- Add one or more findings to a currently-excluded bundle, including the *Excluded Bundle*.
- Delete one or more findings from a previously excluded bundle, including the *Excluded Bundle*.
- Delete an excluded bundle.

An exclusion does not apply to the application immediately when you:

- Add a bundle as an exclusion.
- Add a filter as an exclusion.
- Modify a finding to make it match an excluded filter's criteria.
- Modify a finding so it no longer matches an excluded filter's criteria.

Marking findings as exclusions in a findings table

Procedure

1. Select the finding (or group of findings) in the findings table that may not be important to you or that you do not want to see.
2. Right-click the selection and then choose **Exclude Findings** from the menu. The exclusion or exclusions apply immediately. Excluded findings no longer appear in the table and metrics update immediately.

Results

To view excluded findings, open the Excluded Findings view. Excluded findings also appear in a bundle named **Excluded Bundle**.

To re-include findings that have been excluded, follow the instructions in "Re-including findings that have been marked as exclusions."

Re-including findings that have been marked as exclusions

Excluded findings appear in the Excluded Findings view and in the **Excluded** bundle view. From these views, you can re-include excluded findings.

Procedure

1. In the Excluded Findings view or **Excluded** bundle view, select the finding (or group of findings) that you want to re-include.
2. Right-click the selection and then choose **Include Findings** from the menu.

Results

The included findings are added back to the assessment - and findings tables and metrics are updated immediately to reflect the re-included findings. The findings no longer appear in the Excluded Findings view and in the **Excluded** bundle view.

Example: Specifying filter exclusions

Filter criteria determine if the filter excludes the findings that match or do not match the filter.

These examples describe how to create filters that exclude findings:
- "Example: Filtering and excluding a directory"
- "Example: Filtering and excluding APIs" on page 122

Example: Filtering and excluding a directory

In this example, a filter is created that only shows findings that contain Microsoft `include` files. This filter is then used to narrow down the list of findings (we will exclude all findings that match the filter).

Procedure

1. In the **Directory** section of the Filter Editor view, add the path to the Microsoft include files (for example, `C:\Program Files\Microsoft Visual Studio 8\VC\include`).
2. Select **Restrict to** to make this an inclusive rule.
3. On the Findings view toolbar, click **Show findings that do not match the filter** to see only findings for Microsoft header files. This allows you to see what the scan results will look like after applying the filter's inverse globally and scanning again.
4. Save the filter with a name, such as **MS Includes**.
5. Return to the Configuration perspective and, in the Explorer view, select the C/C++ application or project.
6. If an application is selected, open the open the Exclusions and Filters tab of the Properties view. If a project is selected, open the Filters tab of the Properties view. Click **Add**. Select **MS Includes** and then select **Invert filter**.

7. Save the changes in the Properties view and then scan the application or project again.
8. Return to Triage. The findings in the exclusion appear in the Excluded Findings view.

Example: Filtering and excluding APIs

A common triage scenario might occur early in the triage process when you want to prioritize your findings and there are certain findings that you want to exclude. For example, you determine that three APIs are not threats and you would like to exclude these APIs from subsequent scans.

Procedure

1. In the Filter Editor, in the API section, click **Add** and select three APIs.
2. Select **Restrict to**.
3. Save and name the filter.
4. Return to the Configuration perspective and, in the Explorer view, select the project (or application).
5. In the Properties view, set the behavior of the filter to **Inverted** (in the Select Filter dialog box, select **Invert filter**).
6. Scan again. The APIs in the filter no longer appear in the findings.

Results

Using the same example, you may only want to see the findings that are included in the filter. In this instance, when adding the filter to the list, do not select **Invert filter**. When you scan again, only the findings in the filter appear.

Specifying bundle exclusions from the Properties view

A bundle exclusion eliminates the findings in the bundle. You can only exclude bundles from applications.

Procedure

1. Create a bundle as described in "Creating bundles" on page 123.
2. In the Explorer view, select the application to be associated with the bundle.
3. In the Properties view, select the **Exclusions** tab.
4. Click **Add Bundle** and, in the Select Bundle dialog box, select the bundle that contains the findings to exclude from the application.
5. Click **OK**.
6. Scan again. The bundled findings no longer appear in the findings table.

Triage with bundles

Bundles, which have unique characteristics, may be important to your triage process.

About this task

- Bundles can be exported to defect tracking systems as a single defect or as a defect for each finding in a bundle.
- Bundles can be the basis for report generation.
- Bundles are attached to applications.

Important: A finding can only exist in one bundle at a time. If a finding is in a bundle, moving it to a different bundle will remove it from the first bundle.

This example outlines a simple triage with bundles:

Procedure

1. Scan the source code.
2. Create a bundle named `Resolve ASAP`.
3. Add some critical findings to the bundle.
4. Add notes to the findings in the bundle.
5. Submit the bundle or findings to the defect tracking system - or email them to other developers.
6. Fix the issues.

Creating bundles

Bundle creation occurs in the Bundles view or a view that contains a findings table. You can add findings to an existing bundle or a new bundle.

These topics describe bundle creation in the Bundles and Findings views:
- "Creating a new bundle in the Bundles view"
- "Creating a new bundle in the Findings view"

Note: To be able to create bundles for an assessment, the application that was scanned to create the assessment must be loaded in AppScan Source for Analysis. If you open an assessment for an application that is not loaded, bundle creation actions will not be available.

After you have created one or more bundles, the Findings view **Hide Bundled Findings** action ( ) allows you to toggle the display of bundled findings in the view. This action hides findings in all included bundles that you have created. This setting does not affect the display of findings in excluded bundles - these findings are never shown in the Findings view.

Creating a new bundle in the Bundles view

Procedure

1. In the Bundles view, click **New Bundle** on the toolbar.
2. Name the bundle and click **OK**. The bundle name appears in the Bundles view.
3. To add findings to the bundle, follow the instructions in "Adding findings to existing bundles" on page 124.

Creating a new bundle in the Findings view

Procedure

1. In the Findings view, select findings to add to a bundle.
2. Right-click the selection and choose **Add to Bundle** > **New** in the menu.
3. Name the bundle and click **OK**.

Adding findings to existing bundles

About this task

You can add findings to a bundle from multiple views:
- Findings view
- Excluded Findings view
- Fixed/Modified Findings view
- Missing Findings view
- Report view
- Finding Detail

Tip: You can move a finding from the findings table to the Bundles view using a drag-and-drop operation.

To add findings to a bundle:

Procedure

1. Select the findings that you want to add to the bundle.
2. Right-click the selection and choose **Add to Bundle** > **<bundle name>** (this list contains the five bundles that were most recently created) or **Add to Bundle** > **Select** from the menu.
3. If you choose **Add to Bundle** > **Select**, choose the bundle to add the findings to in the Select Bundle dialog box and then click **OK**.

Moving findings between bundles

Procedure

1. In the Bundles view, open the bundle that contains the finding or findings that you want to move.
2. Select the finding or findings that you want to move and then complete one of these actions:
   - Click **Move to Bundle** or **Move to New Bundle** on the view toolbar. Then select the bundle that you want to move the finding to - or create a new bundle for the finding.
   - Right-click the selection and click **Move to Bundle**. This opens a menu that allows you to select an existing bundle from a list or dialog box - or create a new bundle to move the selection to.

Results

Note: Findings that are moved or added to an excluded bundle will not be excluded in the current assessment. To mark findings as excluded in the current assessment, use the **Exclude Findings** action.

Viewing findings in bundles

When you add findings to a bundle, the findings appear as a row in the bundle. If you open the bundle, you see all findings that the bundle contains.

About this task

Findings from multiple projects in bundles may appear differently. A finding in a bundle appears green and italicized when not found in the most recent scan.

Consider the following example of Application X.

Procedure

1. Application X contains Project A and Project B.
2. Scan Application X.
3. Create a bundle that contains findings from Project A and Project B.
4. Scan Project B. In the Bundles view, findings from Project B appear and findings from Project A appear in green and italics.

Results

A finding highlighted in green and italics is a *fixed/missing finding*. A fixed/missing finding is a finding in a bundle but not in the current assessment. A finding is identified as fixed/missing because it was resolved, removed, or the source file was not scanned. In the Bundles view, the **Excluded** column identifies if the bundle is excluded.

Saving bundles to file

You can save a bundle as a file to open in AppScan Source for Development. A bundle also allows you to import a snapshot of findings from AppScan Source for Analysis to AppScan Source for Remediation.

Procedure

1. Complete one of these actions:
   a. In the Bundles view, select the bundle and click **Save Bundle to File** in the toolbar.
   b. Open the bundle and click **Save Bundle to File** in the toolbar.
2. Select the directory in which to save the bundle file.
3. Name the bundle file (`<file_name>.ozbdl`).

Results

To open a saved bundle:

- In AppScan Source for Development (Eclipse plug-in), select **Security Analysis** > **Open** > **Open Bundle**.
- In AppScan Source for Development (Microsoft Visual Studio plug-in), select **IBM Security AppScan Source** > **Open Bundle**.
- In AppScan Source for Analysis, click **Open Bundle** in the Bundles view toolbar.

Tip: On Windows systems, double-click the bundle file in the Bundles view to open it in AppScan Source for Analysis or AppScan Source for Development.

Submitting bundles to defect tracking and by email

The findings in bundles can be submitted to your corporate defect tracking system - or sent by email. Once you place findings in a bundle, you can submit these findings as bugs for developer remediation.

Procedure

1. Open the bundle.
2. Click the **Submit bundle to defect tracking** toolbar button down arrow and then select your defect tracking system.

   Note: Depending on your defect tracking system, you may want to modify Defect Tracking System preferences before submitting the bundle.

   Alternatively, on the Bundle toolbar, click **Email Bundle** to send the bundle to others (email preferences must be configured beforehand).
3. Complete the configuration dialog boxes that open. These vary depending on the defect tracking system that you have chosen - and are described in the *AppScan Source for Analysis and defect tracking section* of the help.

Adding notes to bundles

Procedure

1. In the Bundles view, select the bundle to annotate.
2. Click **Add Notes** on the Bundle toolbar or right-click the selection and choose **Add Notes** in the menu.
3. Type the note and click **OK**.

Modifying findings

Modified findings are findings that have changed vulnerability types, classifications, or severities - or that have annotations. The Modified Findings view displays these findings for the current application (the application that is active as a result of opening an assessment for it). In the My Assessments view (available only in AppScan Source for Analysis), the **Modified** column indicates if a finding has changed in the current assessment.

Modifications to findings are immediate and cause metrics to update. Modifications are stored with applications - and are applied to their future scans.

Findings can be modified in the Finding Detail view or from any view with a findings table. The Finding Detail view allows modification of individual findings, or you can modify multiple findings in a findings table.

Note: You must have **Save Assessments** permission to save changes after modifying an assessment.

Making modifications from a findings table

You may want to modify findings via a findings table if you will be making the same changes to multiple files. If you will be modifying an individual finding, use a findings table or the Finding Detail view.

- “Changing the vulnerability type”
- “Promoting finding classifications” on page 127
- “Modifying severity” on page 127
- “Supported annotations and attributes” on page 137

Changing the vulnerability type

Vulnerability types can be changed for individual findings or a group of findings.

Procedure

1. From a findings table, select a finding or group of findings to modify.
2. Right-click the selection and choose **Set Vulnerability Type** from the menu.
3. In the Select Vulnerability Type dialog box, choose the vulnerability type that you want and click **OK**.

Promoting finding classifications

A finding with a classification of suspect security finding or scan coverage finding can be promoted to a definitive finding.

Procedure

1. From a findings table, select a finding or group of findings to modify.
2. Right-click the selection and choose **Promote to Definitive** from the menu.

Modifying severity

Selecting a new severity level changes the severity for each selected finding. For example, AppScan Source might report that an API is of medium severity, but your corporate policy identifies it as more severe. You can modify the severity to meet your requirements, but note that AppScan Source remediation assistance does not contain the modification.

Procedure

1. From a findings table, select a finding or group of findings to modify.
2. Right-click the selection and choose **Set Severity** in the menu.
3. Select **High**, **Medium**, **Low**, or **Info** as the new severity level.

Annotating findings

Notes can be used as reminders for you to take further action on a finding - or to convey information about the finding to someone else. You can add a note to a single finding or to a group of findings.

Procedure

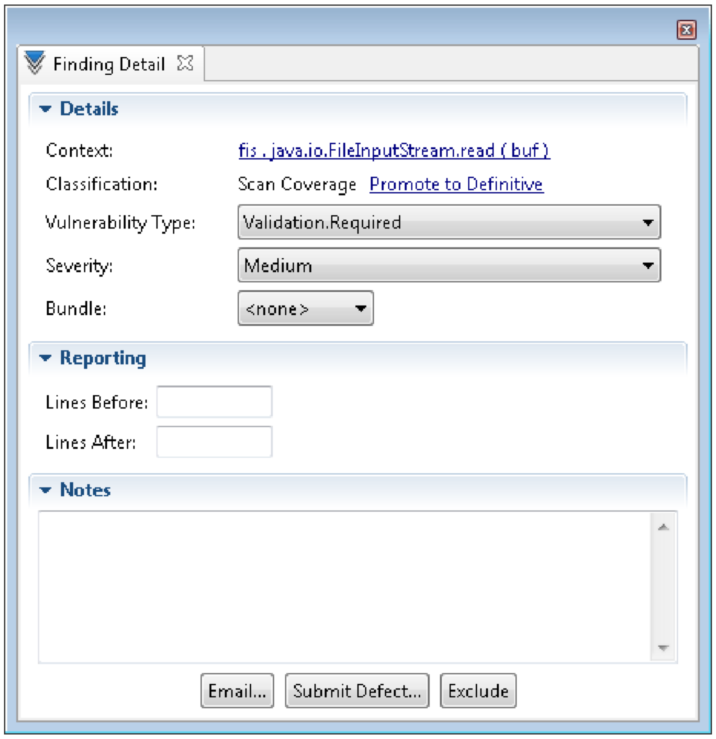
1. From a findings table, select a finding or group of findings to modify.
2. Right-click the selection and choose **Add Notes** from the menu.
3. Type the note and then click **OK**.

Modifying findings in the Finding Detail view

Individual findings can be modified in the Finding Detail view. If you select a finding in a table and open the Finding Detail view, the selected finding and its characteristics appear.

Finding Detail view

When you select a finding, the Finding Detail view displays and allows you to modify its properties. With this view, you can modify an individual finding.



- “Details section”
- “Reporting section (available in AppScan Source for Analysis and AppScan Source for Development (Eclipse plug-in) only)”
- “Notes Section”
- “Finding Detail view actions” on page 129
- “Finding Detail view for custom findings (available in AppScan Source for Analysis only)” on page 129

Details section

- **Context**: Snippet of code that surrounds the vulnerability
- **Classification**: Definitive or Suspect security findings - or Scan Coverage findings - with a link to promote the finding to **Definitive** or revert to the original value if the classification was changed
- **Vulnerability Type**
- **Severity**: High, medium, low, or info
- **Bundle**: Name of the bundle that contains the findings (not available in AppScan Source for Development (Visual Studio plug-in))

Reporting section (available in AppScan Source for Analysis and AppScan Source for Development (Eclipse plug-in) only)

Specify the number of lines of code to include before and/or after the finding in reports.

Notes Section

Annotate the finding.

Finding Detail view actions

- **Exclude**: Click **Exclude** to exclude (remove) the finding from the findings table. To view excluded findings, open the Excluded Findings view.
- Available in AppScan Source for Analysis only:
  - **Email**: If you have configured email preferences, you can email a finding bundle directly to developers to advise them of potential defects found after a scan. The email includes a bundle attachment that contains the findings, and the email text describes the findings.
    1. To email the current finding in the Finding Detail view, click **Email**.
    2. In the Attachment File Name dialog box, specify a name for the finding bundle that will be attached to the email. For example, specifying `my_finding` in the **Attachment File Name** field causes a bundle with file name `my_finding.ozbdl` to be attached to the email.
    3. Click **OK** to open the Email Findings dialog box. By default, the **Mail To** field in the Email Findings dialog box will populate with the **To Address** that is specified in the email preferences - however, it can easily be changed when preparing the email. In this dialog box, review the contents of the email and then click **OK** to send the email.
  - **Submit Defect**: To submit the finding as a defect, click **Submit Defect**. This opens the Select Defect Tracking System dialog box.
    - If you select **ClearQuest** and click **OK**, the Attachment File Name dialog box opens. In it, specify a name for the finding bundle that will be attached to the defect and then click **OK**. Log in to Rational ClearQuest and submit the findings.
    - If you select **Quality Center** and click **OK**, the Login dialog box opens, allowing you to log in to Quality Center to submit the findings.
    - If you select either **Team Foundation Server** option, a dialog box opens, prompting you to log into the defect tracking system and provide other configuration details.

    Note: Rational Team Concert is the only supported defect tracking system on OS X.

Finding Detail view for custom findings (available in AppScan Source for Analysis only)

The Finding Detail view for custom findings provides additional information that you can edit:

- File
- Line
- Column
- API

In addition, the method by which you edit the “Details section” on page 128 is different than standard findings for some fields (for example, the classifications for custom findings appear in a list).

Removing finding modifications

If you have modified findings, you can remove the modifications (revert back to original values) using the methods described in this topic.

About this task

There are a variety of methods for removing finding modifications:

- "Removing modifications in the Modified Findings view": This method requires that an assessment is open for the application that contains modifications that you want to remove. It is useful when you want to revert multiple modified findings.
- "Removing modifications in other views with findings": This method requires an open assessment - and is particularly useful if you have made more than one modification to a finding, and you want to revert a subset of the changes. For example, if you have changed the severity and classification of a finding - and want to revert to the original severity while keeping the modified classification - this method is most suitable.
- "Removing modifications in the Modified Findings tab of the Properties view (AppScan Source for Analysis only)": This method is useful if you want to remove modifications for an application that does not have an open assessment - and can be used for reverting multiple modified findings.

Removing modifications in the Modified Findings view

Procedure

1. In the Modified Findings view, select the modified finding that you want to revert. Multiple findings can be selected using the keyboard Ctrl and Shift keys on Windows - or the command and shift keys on OS X.
2. Click **Delete Modifications** or right click the selection and choose **Delete Modifications** from the menu.

Results

This action removes all modifications that have been made to a finding. If you have made more than one modification to a finding, and you want to revert a subset of the changes, use the method described in "Removing modifications in other views with findings."

Removing modifications in other views with findings

About this task

In any view that contains a findings table, you can choose the columns to display using the **Select and Order Columns** action. Using this feature, you can display **Severity (Original)**, **Severity (Custom)**, **Classification (Original)**, and **Classification (Custom)** columns. These columns assist you in returning modifications to their original values (by actions in the findings table or by using the Finding Detail view). For example, given a finding with a **Severity** or **Severity (Custom)** value of **High** - and a **Severity (Original)** value of **Medium** - the severity level can be returned to **Medium** using a variety of methods, such as:

- In a findings table, right click the finding and choose **Set Severity** > **Medium** in the menu.
- Select the finding and then, in the Finding Detail view, set the **Severity** field to **Medium**.

Removing modifications in the Modified Findings tab of the Properties view (AppScan Source for Analysis only)

Procedure

1. In the Explorer view, select the application that contains the modifications that you want to remove.

2. In the Modified Findings view, select the modified finding that you want to revert. Multiple findings can be selected using the keyboard Ctrl and Shift keys.
3. Click **Delete Modifications** or right click the selection and choose **Delete Modifications** from the menu.

Results

This action removes all modifications that have been made to a finding. If you have made more than one modification to a finding, and you want to revert a subset of the changes, use the method described in “Removing modifications in other views with findings” on page 130.

Comparing findings

Assessments are compared using the **Diff Assessments** action. When two assessments are compared, the differences between the two are displayed in the Assessment Diff view. This view displays new, fixed/missing, and common findings.

These controls are available in the Assessment Diff view:
- **Diff Assessments**: Display the differences between the two selected assessments.
- **New Findings** (blue): Use this toolbar button to toggle the display of new findings (findings that are in the blue-labelled assessment, but not the green-labelled assessment).
- **Fixed/Missing Findings** (green): Use this toolbar button to toggle the display of fixed/missing findings (findings that are in the green-labelled assessment, but not the blue-labelled assessment).
- **Common** (white): Use this toolbar button to toggle the display of findings that are common between the two assessments.
- **Next**: Move to the next block of new or fixed/missing findings.
- **Previous**: Move to the previous block of new or fixed/missing findings

Comparing two assessments in the Assessment Diff view

Procedure

1. In the left pane, select two assessments to compare.
2. Click the **Diff Assessments** toolbar button or right-click the selection and choose **Diff Assessments** from the menu.

Comparing two assessments from the main menu bar

Procedure

1. Select **Tools** > **Diff Assessments** in the main menu bar.
2. In the Diff Assessments dialog box, select two assessments.
3. Click **OK** to open a comparison of the two assessments in the Assessment Diff view.

Finding differences between assessments in the My Assessments and Published Assessments views

Procedure

1. Select two assessments in one of the views.
2. Click the **Diff Assessments** toolbar button or right-click the selection and then choose **Diff Assessments** from the menu. This opens a comparison of the two assessments in the Assessment Diff view.

Custom findings

To augment your analysis results, you can create *custom findings*. These are user-created findings that AppScan Source for Analysis adds to the currently-open assessment or selected application. Custom findings impact assessment metrics and can be included in reports. Once created, a custom finding is automatically included in future scans of the application.

The behavior of a custom finding depends on the view from which it is created.

When created from the Findings view, the custom finding:
- Is applied to the currently-open assessment.
- Is saved as part of the application and appears in the application properties.
- Affects the current scan and future scans of the same application.
- Affects assessment metrics immediately.

When created from the Properties view or by selecting the **Add Custom Finding** action for a selected application, the custom finding:
- Is applied to the selected application.
- Is added to the current assessment if the application is the application that was scanned.
- Is contained in future scans of that application.

When created from the code editor:
- If an assessment is open, the custom finding operates as when created in the Findings view.
- If no assessment is open, the custom finding operates as when created in the Properties view.

AppScan Source for Analysis automatically saves the application after you create custom findings. You cannot modify the assessment without modifying the application. However, if an assessment is not associated with an application, no application is modified.

If you add custom findings to an application, they are included in subsequent scans of that application and cannot be excluded. To remove a custom finding, you must exclude it from an assessment or delete it from the application.

Note: Custom findings cannot be *fixed/missing*.

A custom finding consists of these attributes:
- **Vulnerability Type** (required)
- **Severity** (required)

- **Classification** (required)
- **File** (required)
- **Context**
- **Line** number
- **Column** number
- **API**
- **Notes**
- **Bundle**

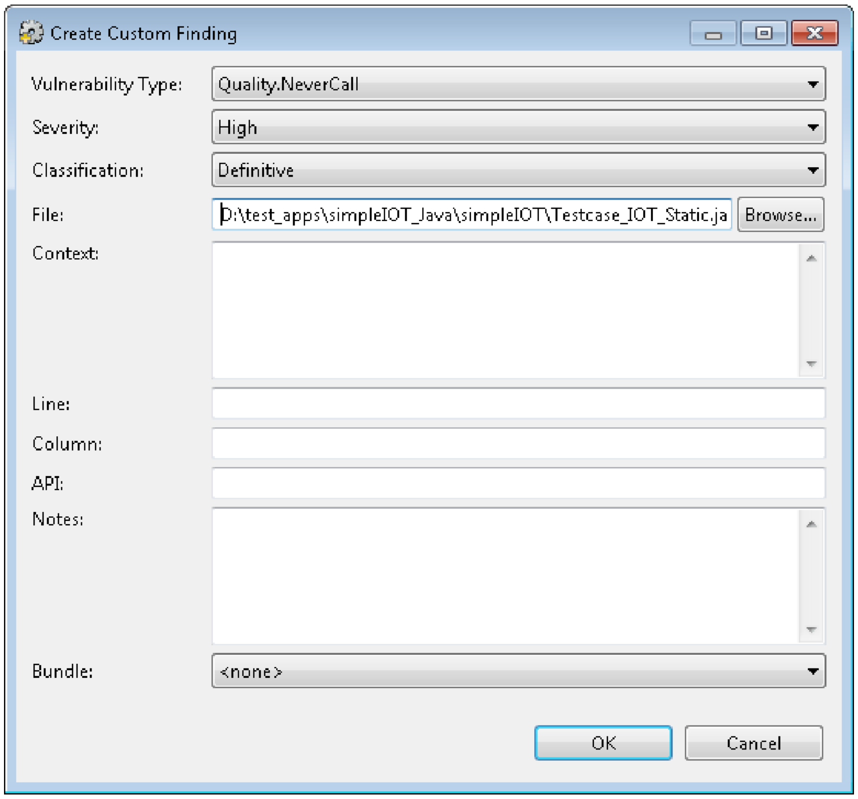
Creating a custom finding in the Properties view

Creating or editing a custom finding from the application Properties view affects current assessment results and future scans.

Procedure

1. Select the application in the Explorer view.
2. In the Properties view, select the **Custom Findings** tab.
3. Click **Create Custom Finding** on the toolbar.
4. In the Create Custom Finding dialog box, add the required items:
   - **Vulnerability Type**
   - **Severity**
   - **Classification**
   - **File**

   Optionally add context, line number, column, API, notes, and bundle designation.


Create Custom Finding

Vulnerability Type: Quality.NeverCall
Severity: High
Classification: Definitive
File: D:\test_apps\simpleIOT_Java\simpleIOT\Testcase_IOT_Static.ja
Context:
Line:
Column:
API:
Notes:
Bundle: <none>

5. Click **OK** to save the custom findings to the application.

Modifying or removing a custom finding in the Properties view

Creating or editing a custom finding from the application Properties view affects current assessment results and future scans.

Procedure

1. Select the finding. If you are removing custom findings, you can select a group of findings to delete.
2. To modify the custom finding, click **Edit the selected finding** on the toolbar - and then modify the previously-defined finding information.
3. To remove the custom finding or findings, click **Delete the selected findings** on the toolbar.

Creating custom findings in a findings view

You can create or manage custom findings from multiple findings views (for example, the Findings and Custom Findings views).

Creating a custom finding from a view adds the new finding to the current assessment and updates the assessment metrics.

When adding a custom finding in a findings view, click the view's **Create Custom Finding** toolbar button. This opens the Create Custom Finding dialog box, which you complete in the same manner as described in "Creating a custom finding in the Properties view" on page 133.

To remove a custom finding, you must exclude it from an assessment or delete it from the application - or by following the instructions in "Modifying or removing a custom finding in the Properties view." These actions are not available in other findings views.

Creating custom findings in the source code editor

About this task

When you add custom findings using the source code editor, these conditions apply:

- If the visible source file in the source code editor belongs to the currently-opened assessment, the custom finding is added to the assessment and the associated application.
- If the custom finding does not belong to the currently-opened assessment, the custom finding is only added to the application that contains the source file.
- If the source file belongs to more than one application, or AppScan Source for Analysis cannot determine the application, you must select the appropriate application.

If you create a custom finding from the source code editor, the Create Custom Finding dialog box pre-populates with information from the editor.

- **File**: Name of the currently-opened file
- **Context**: Any selected text in the editor. If text is not selected, context is the current line of the cursor location. If multiple lines are selected, all selected lines become the context.
- **Line** number and **column** number: Current line and column number

To create a custom finding from the editor:

Procedure

1. Select the lines of code to add as a custom finding.
2. Right-click the selection and choose **Create Custom Finding** from the menu. The Create Custom Finding dialog box populates with the file, context, column number, and line number.
3. Select the **Vulnerability Type**, **Severity**, and **Classification**. Optionally add an API, notes, or bundle designation.
4. Click **OK**.

Resolving security issues and viewing remediation assistance

AppScan Source alerts you to security errors or common design flaws and assists in the resolution process. The AppScan Source Security Knowledgebase - and internal or external code editors - help with this process.

About this task

The AppScan Source Security Knowledgebase offers suggestions for correcting findings. This in-context intelligence for each vulnerability offers precise descriptions about the root cause, severity of risk, and actionable remediation advice. For example, it describes `strcpy()`, a Buffer Overflow type, as having a high severity level and provides this remediation assistance:

> `strcpy` is susceptible to destination buffer overflow because it does not know the length of the destination buffer and therefore cannot check to make sure it does not overwrite it. You should consider using `strncpy` that takes a length parameter. `strncpy` is a security risk as well, although to a lesser degree.

To view the AppScan Source Security Knowledgebase:

Procedure

- In AppScan Source for Analysis, open the Remediation Assistance view and then select a finding in the findings table. Remediation assistance for that particular finding displays. Alternately, select **Help** > **Security Knowledgebase** from the main menu bar to open the entire AppScan Source Security Knowledgebase in a browser.
- In AppScan Source for Development (Eclipse plug-in), open the Remediation Assistance view and then select a finding in the findings table. Remediation assistance for that particular finding displays.
- In AppScan Source for Development (Visual Studio plug-in), select a finding in a findings table. Select **IBM Security AppScan Source** > **Knowledgebase Help** from the main menu bar - or right-click the finding and select **Knowledgebase Help** from the menu. This opens the remediation assistance for the selected finding.

Analyzing source code in an editor

With AppScan Source, you can analyze or modify source code in an internal editor - or you can choose from a variety of external editors.

External editors allow you to review results in AppScan Source for Analysis and make code modifications in the development environment of your choice. External editors include:

Table 13. Supported external editors

Editor	**Platform**
Eclipse (see the AppScan Source system requirements to learn which versions of Eclipse are supported)	Windows and Linux
Notepad	Windows
vi	Linux
System Default	Windows and Linux

Note: You cannot edit source files in a `WAR` file.

To view/modify source code in the editor, choose one of these options:

- Double-click a finding in the findings table. The internal editor opens at the line of code.
- Right-click a finding in the findings table and select **Open in Internal Editor** or **Open in External Editor** > **<editor>** (where **<editor>** is one of the supported external editors listed in the above table).
- Select a trace node and then select the **Open in Internal Editor** or **Open in External Editor** > **<editor>** toolbar button - or right-click the selection and select **Open in Internal Editor** or **Open in External Editor** > **<editor>** from the menu.

If you have opened a file in the editor, markers indicate locations in the file that represent findings. To follow these back to the findings table, right-click the line of code in the editor and then select **Show in Findings View** from the menu.

Resolving quality issues

If you open a scan that was run in a product that supports quality analysis, quality-related findings will appear in findings tables. F1 Help is available for each type of finding. In most cases, the quality issue is explained by the help and examples and sample solutions are offered. This information can greatly assist in resolving quality issues.

About this task

Scanning for quality issues is supported in AppScan Source for Development (Eclipse plug-in) (Java only), the AppScan Source command line interface (CLI) (Java and C/C++), and AppScan Source for Automation (Java and C/C++). Saved scans that include quality findings can be opened in these and other supported AppScan Source products.

Note:

- Findings that are classified with the **Info** severity level are not included in the Vulnerability Matrix view.
- If you open a scan that was run in a product that supports quality analysis, findings related to quality are not reflected in the Vulnerability Matrix view.

Procedure

1. Select a finding in the findings table and press the **F1** keyboard key.
2. Help for the finding's quality issue opens in the Help view. In most cases, links to examples and solutions for the quality issue is available in the help. This help may assist you in resolving the quality issue.

3. If your quality scan includes Java or C/C++ software metrics rules, you can use the Quality Metrics view to assist in issue resolution. This view offers a read-only compact view of analysis results organized by metric category with F1 Help for each metric entry.

Supported annotations and attributes

Some annotations or attributes that are used to *decorate* code are processed during scans. When a supported annotation or attribute is found in your code during a scan, the information is used to mark the decorated method as a tainted callback. A method marked as a tainted callback is treated as if all of its arguments have tainted data. This results in more findings with traces. Supported annotations and attributes are listed in this help topic.

Supported Java annotations

Note: Java annotations are only supported in AppScan Source for Analysis and AppScan Source for Development (Eclipse plug-in).

Table 14. Supported Java annotations

Annotation	Abbreviation
`javax.xml.ws.WebServiceProvider`	`@WebServiceProvider`
`javax.jws.WebService`	`@WebService`
`javax.jws.WebMethod`	`@WebMethod`

Supported Microsoft .NET attributes

Note: Microsoft .NET annotations are only supported in AppScan Source for Analysis and AppScan Source for Development (Visual Studio plug-in)

Table 15. Supported Microsoft .NET attributes

Attribute	Abbreviation
`System.Web.Services.WebServiceAttribute`	`WebService`
`System.Web.Services.WebMethodAttribute`	`WebMethod`

Chapter 6. AppScan Source trace

With AppScan Source trace, you can verify input validation and encoding that meets your software security policies. You can look at the findings that produce input/output traces and mark methods as validation and encoding routines, sources or sinks, callbacks, or taint propagators.

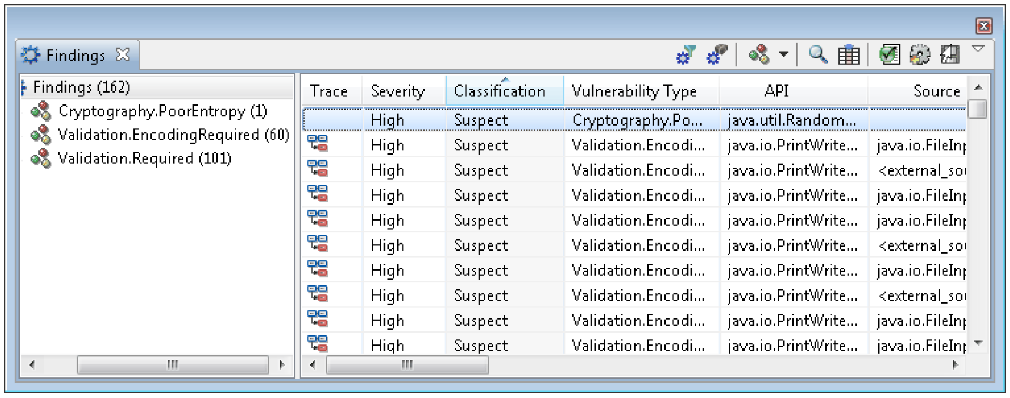
AppScan Source traces the flow of data through an application, across modules and languages. It displays the paths of potentially dangerous data in a call graph, indicating areas where an application may be susceptible to vulnerabilities.

Tracing helps you defeat SQL Injection, cross-site scripting, and other input validation attacks by identifying the lack of approved input validation and encoding routines in applications. You interactively trace the entire call graph, clicking directly from the Trace view to see the source in the development environment or code editor of your choice. Tracing also enables policy enforcement, allowing you to identify approved routines required for proper input validation and encoding, taint propagation, or sinks and sources, and include them in future scans.

When a scan results in a trace, you can create input validation or encoding routines, vulnerabilities, sinks, sources, or taint propagators for specific findings from the Trace view. For example, if you mark a routine as a validation routine in AppScan Source for Analysis and add it to the AppScan Source Security Knowledgebase, subsequent scans no longer report `Validation.Required` or `Validation.Encoding.Required` findings for data paths on which the routines are called. In the Trace view, you can also define vulnerabilities as a source, sink, or both - and identify a method as a taint propagator, a tainted callback, or not being susceptible to taint.

AppScan Source trace scan results

Scan results may include traces identified by AppScan Source trace. The icon in the **Trace** column indicates the existence of a trace of the call graph.



- Findings (162)
  - Cryptography.PoorEntropy (1)
  - Validation.EncodingRequired (60)
  - Validation.Required (101)

Trace	Severity	Classification	Vulnerability Type	API	Source
	High	Suspect	Cryptography.Po...	java.util.Random...	
▣	High	Suspect	Validation.Encodi...	java.io.PrintWrite...	java.io.FileInp
▣	High	Suspect	Validation.Encodi...	java.io.PrintWrite...	<external_sou
▣	High	Suspect	Validation.Encodi...	java.io.PrintWrite...	java.io.FileInp
▣	High	Suspect	Validation.Encodi...	java.io.PrintWrite...	java.io.FileInp
▣	High	Suspect	Validation.Encodi...	java.io.PrintWrite...	<external_sou
▣	High	Suspect	Validation.Encodi...	java.io.PrintWrite...	java.io.FileInp
▣	High	Suspect	Validation.Encodi...	java.io.PrintWrite...	<external_sou
▣	High	Suspect	Validation.Encodi...	java.io.PrintWrite...	java.io.FileInp
▣	High	Suspect	Validation.Encodi...	java.io.PrintWrite...	java.io.FileInp

Scans may generate findings of type `Validation.Required` and `Validation.EncodingRequired`. These findings indicate a location in the source code where data is read from an external source or saved to an external sink. The scan flags these cases because the data should be validated or encoded to prevent malicious or erroneous data from doing harm.

Validation and encoding

Validation is the process of checking input data to ensure that it is well-formed. A `Validation.Required` finding indicates that no validation occurred along a given data path from source to sink. Validation can be as simple as bounding the data to a maximum length and as complex as checking for well-formed names and addresses. Validation can also check for attacks such as SQL Injection by detecting illegal character sequences that enable these attacks.

Encoding is the process of transforming the data into a well-formed state. A `Validation.EncodingRequired` finding indicates that no encoding occurred along a given data path from source to sink. Encoding could be as simple as escaping characters or as complex as encrypting the data. Encoding can also prevent attacks such as Cross-Site Scripting by escaping the characters that lead to these attacks.

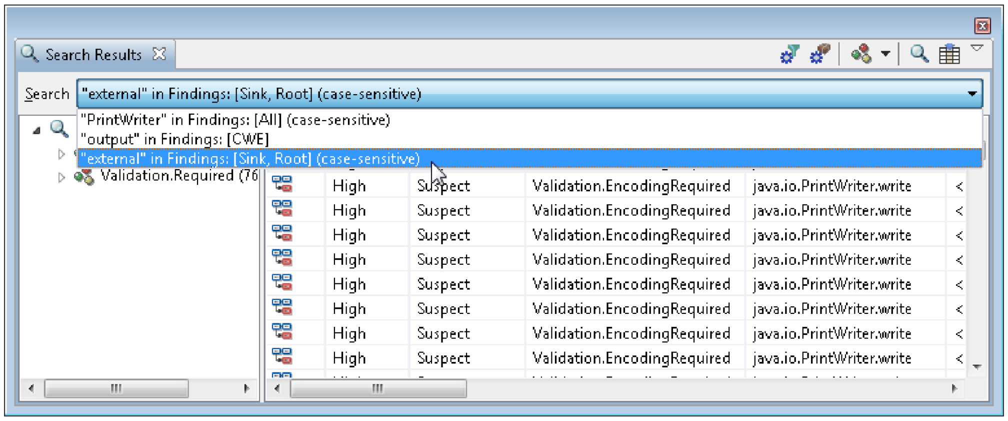
When you first scan, AppScan Source may identify a finding as a suspect security finding. When you create a validation or encoding routine that applies to a specific source, AppScan Source for Analysis reports the finding as definitive (instead of suspect) if the specified validation or encoding routine is not called after it receives data from the source.

Assessments track data from known sources throughout a project. If data can be tracked from a known source to a known sink, specified validation and encoding routines can ensure that a malicious attack could not occur with unbounded input data.

Searching AppScan Source traces

If you want to group trace findings, you can search for sources or sinks. This causes the trace findings to appear in the Search Results view.

In the Trace view, click **Search for traces with the same type routine**. Then, in the Search Findings dialog box, select source, sink, lost sink (includes virtual lost sinks), virtual lost sink, or trace calls to isolate the results to the traces that contain the string. The search results, which are cumulative, appear in the Search Results view. From this view you can search again to refine the search.



Search: "external" in Findings: [Sink, Root] (case-sensitive)

"PrintWriter" in Findings: [All] (case-sensitive)
"output" in Findings: [CWE]
"external" in Findings: [Sink, Root] (case-sensitive)

	High	Suspect	Validation.EncodingRequired	java.io.PrintWriter.write	<
	High	Suspect	Validation.EncodingRequired	java.io.PrintWriter.write	<
	High	Suspect	Validation.EncodingRequired	java.io.PrintWriter.write	<
	High	Suspect	Validation.EncodingRequired	java.io.PrintWriter.write	<
	High	Suspect	Validation.EncodingRequired	java.io.PrintWriter.write	<
	High	Suspect	Validation.EncodingRequired	java.io.PrintWriter.write	<
	High	Suspect	Validation.EncodingRequired	java.io.PrintWriter.write	<
	High	Suspect	Validation.EncodingRequired	java.io.PrintWriter.write	<

Input/output tracing

An *input/output trace* is generated when AppScan Source for Analysis can track the data from a known source to a *sink* or *lost sink*.

Input/Output Trace

If the code analysis can track a tainted source to a sink or lost sink, then the analysis produces an input/output trace. The root of the trace is the method that gets data from taint-producing sources and passes it to a series of calls that eventually write to an unprotected sink.

Sources and sinks

- **Source**: A source is an input to the program, such as a file, servlet request, console input, or socket. For most input sources, the data returned is unbounded in terms of content and length. When an input is unchecked, it is considered tainted. Sources are listed in any findings table in the **Source** column.
- **Sink**: A sink can be any external format to which data can be written out. Sink examples include databases, files, console output, and sockets. Writing data to a sink without checking it may indicate a serious security vulnerability.
- **Lost Sink** A lost sink is an API method that can no longer be traced.

  Note: Lost sinks do not apply to JavaScript findings.

Using the Trace view

About this task

In the Trace view, you view a single input/output trace for a finding. The pane is divided into three panels:
- Input and output stacks
- Data Flow
- Graphical Call Graph

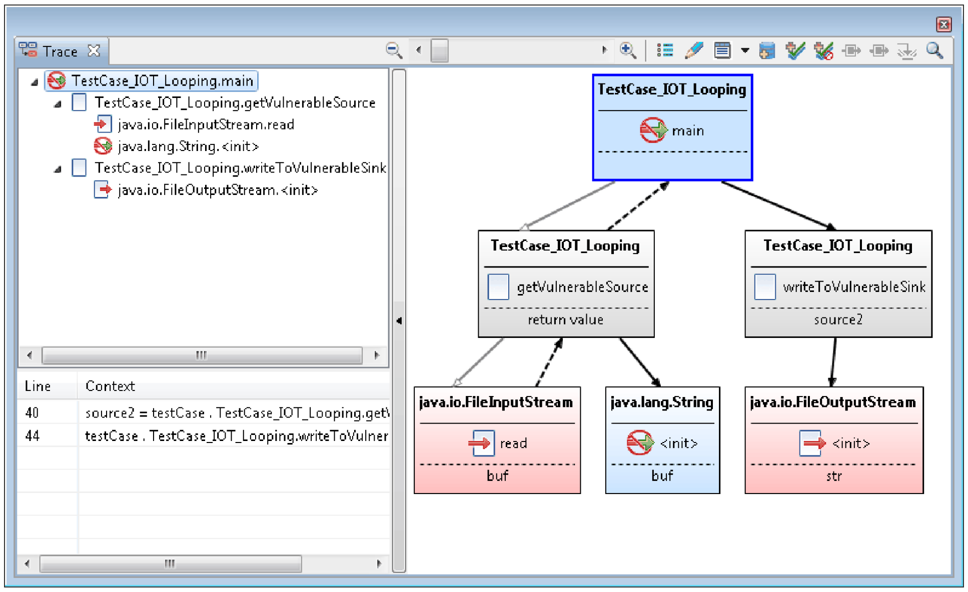
These panels are described in greater detail in “Input/output stacks in the Trace view” on page 142.

Note: In a JavaScript trace, a “JavaScript Statement Graph” on page 143 is displayed rather than a Graphical Call Graph.

To view an AppScan Source trace:

Procedure

1. Scan and find trace results in the Findings view.
2. From the View menu, open the Trace view.
3. Select the rows in the findings table that display the **Trace** icon. The Trace view displays the trace details.



- TestCase_IOT_Looping.main
  - TestCase_IOT_Looping.getVulnerableSource
    - java.io.FileInputStream.read
    - java.lang.String.<init>
  - TestCase_IOT_Looping.writeToVulnerableSink
    - java.io.FileOutputStream.<init>

Line	Context
40	source2 = testCase . TestCase_IOT_Looping.getV
44	testCase . TestCase_IOT_Looping.writeToVulner

Input/output stacks in the Trace view

The upper left panel displays the **input** and **output** stacks. The stack is a sequence of calls that terminates at either a source (input stack) or sink (output stack).

Data Flow

The lower left panel contains the data flow for the selected method. Data can flow through a method call or an assignment. The data flow section displays the line number in the source code where the item and context appear.

Call Graph

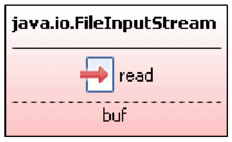
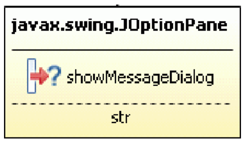
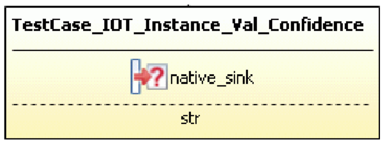
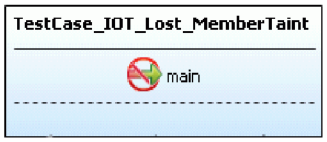
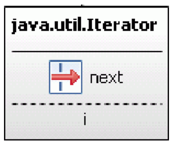
Note: In a JavaScript trace, a "JavaScript Statement Graph" on page 143 is displayed rather than a Graphical Call Graph.

The chart is a graphical representation of the call graph. Each method call is a rectangle within the graph showing the class name and the method name:

- Red identifies the method call as a source, sink, or both.
- A *lost sink* is an API method that can no longer be traced. A *virtual lost sink* is a lost sink that is also a virtual function (a function that can have more than one implementation). Yellow identifies the method call as a lost sink or virtual lost sink.
- Blue indicates that the method call is not a validation/encoding routine.
- Grey represents all other trace node types.

Each method call is divided into three sections: the class name, the method name, and the tainted argument name. Hover text for the method call provides greater detail.

Lines with arrows represent calls from method to method. An unfilled arrowhead indicates that there was no known tainted data in the call, while a solid arrow indicates tainted data flow. A dashed arrow indicates a return statement.

Symbol	Description
——▷	Method call with no known tainted data
——▶	Method call with tainted data
- - - -▶	Return with tainted data
	Source or sink (red): • A source is a method, function, or parameter that is the origin of potentially untrustworthy data. • A sink is a method or function that is potentially vulnerable to tainted data or is potentially dangerous to use.
	Lost sink (yellow): A method/function that is potentially vulnerable to tainted data or is potentially dangerous to use.
	Virtual lost sink (yellow): A type of lost sink that is resolved to more than one concrete implementation.
	Not a validation/encoding routine (blue). Marking an API as *not a validation/encoding routine* identifies that this API does not validate any data.
	Taint propagator: A function/method that propagates taint to one or more of its parameters, to its return value, or to this pointer.

Tip:

- In the Trace view, hovering over trace nodes in the graph provides information about the node.
- The two left panels in the view (the input/output stacks panel and the data flow panel) can be collapsed for easier viewing of the graphical call graph. To collapse these panels, select the **Hide tree view** arrow button. To display these panels when they are hidden, select the **Show tree view** arrow button.
- Move the scroll bar to zoom in and focus on details - or to zoom out to see more. Hovering over the zoom scroll bar provides the current zoom level. To zoom in to the maximum level, click **Zoom to 200%**. To zoom out as far as possible, click **Zoom to fit**.

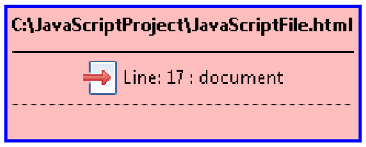
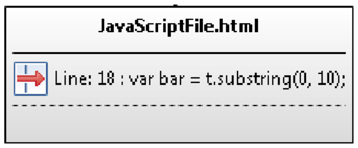
JavaScript Statement Graph

The statement graph section of a JavaScript trace displays the data flow between statements.

Within the graph, each statement is a rectangle that provides the following information:

- The path and file name of the affected file. If the next statement is located in the same file, only the file name is listed.
- The line number that contains the statement.
- If available, the section of code that is of interest.
- If the rectangle is red, the statement is a source, sink, or both.
- If the rectangle is grey, the statement is a taint propagator.
- Hover text for the statement provides greater detail.

Lines with arrows represent data flowing from statement to statement.

Symbol	Description
→	Flow of tainted data
 C:\JavaScriptProject\JavaScriptFile.html Line: 17 : document	Source or sink (red): • A source is a statement that is the origin of potentially untrustworthy data. • A sink is a statement that is potentially vulnerable to tainted data or is potentially dangerous to use.
 JavaScriptFile.html Line: 18 : var bar = t.substring(0, 10);	Taint propagator: A statement that propagates taint to one or more of its parameters, to its return value, or to this pointer.

Tip:

- In the Trace view, hovering over trace nodes in the graph provides information about the node.
- The two left panels in the view (the input/output stacks panel and the data flow panel) can be collapsed for easier viewing of the graphical call graph. To collapse these panels, select the **Hide tree view** arrow button. To display these panels when they are hidden, select the **Show tree view** arrow button.
- Move the scroll bar to zoom in and focus on details - or to zoom out to see more. Hovering over the zoom scroll bar provides the current zoom level. To zoom in to the maximum level, click **Zoom to 200%**. To zoom out as far as possible, click **Zoom to fit**.

Analyzing source code in an editor

With AppScan Source, you can analyze or modify source code in an internal editor - or you can choose from a variety of external editors.

External editors allow you to review results in AppScan Source for Analysis and make code modifications in the development environment of your choice. External editors include:

Table 16. Supported external editors

Editor	Platform
Eclipse (see the AppScan Source system requirements to learn which versions of Eclipse are supported)	Windows and Linux
Notepad	Windows

Table 16. Supported external editors (continued)

Editor	Platform
vi	Linux
System Default	Windows and Linux

Note: You cannot edit source files in a `WAR` file.

To view/modify source code in the editor, choose one of these options:
- Double-click a finding in the findings table. The internal editor opens at the line of code.
- Right-click a finding in the findings table and select **Open in Internal Editor** or **Open in External Editor** > **<editor>** (where **<editor>** is one of the supported external editors listed in the above table).
- Select a trace node and then select the **Open in Internal Editor** or **Open in External Editor** > **<editor>** toolbar button - or right-click the selection and select **Open in Internal Editor** or **Open in External Editor** > **<editor>** from the menu.

If you have opened a file in the editor, markers indicate locations in the file that represent findings. To follow these back to the findings table, right-click the line of code in the editor and then select **Show in Findings View** from the menu.

Validation and encoding scope

From the Trace view, you can specify custom validation and encoding routines that, once stored in the AppScan Source Security Knowledgebase, marks data as checked instead of tainted. With the Custom Rules Wizard, you define these routines based on their scope.

See "Example 4: Validation in depth" on page 155 for the procedure to create validation and encoding routines.

Validation or encoding routines are based upon their scope and are defined as:
- "API specific"
- "Call site specific"

API specific

API specific validation and encoding routines may be associated with a single project or multiple projects.

API specific routines will untaint any data coming from all instances of a specific source API. For example, you could specify a validation routine for any input from the API:

```
javax.servlet.ServletRequest.getParameter
(java.lang.string):java.lang.string
```

API specific routines are stored on the server. API specific routines for a project are stored in the project.

Call site specific

Call site specific routines are always associated with a single project.

Call site specific routines will untaint data coming from a specific location in the code. When you create a call site specific validation or encoding routine, you specify that the routine applies to a particular input call site. Call site specific routines are always stored in the project.

Note: Call site specific applies to any call to the validation routine within the same method.

Creating custom rules from an AppScan Source trace

You can create custom rules from the Trace view that allow you to filter out findings with traces that are taint propagators, not susceptible to taint, or sinks. You can also mark methods in the trace as validation/encoding routines (or indicate that they are not validation/encoding routines).

About this task

See "Example 2: Creating a Validation/Encoding Routine from the Trace view" on page 150 for an example of source code, the output, and the procedure to create the validation and encoding routines.

Table 17. Valid markings for Trace view nodes

Selected method	Valid marking
Intermediary nodes	• Validation/encoding routines • Not susceptible to taint • Not a validation/encoding routine
Lost sink	• Taint propagator • Not susceptible to taint • Sink

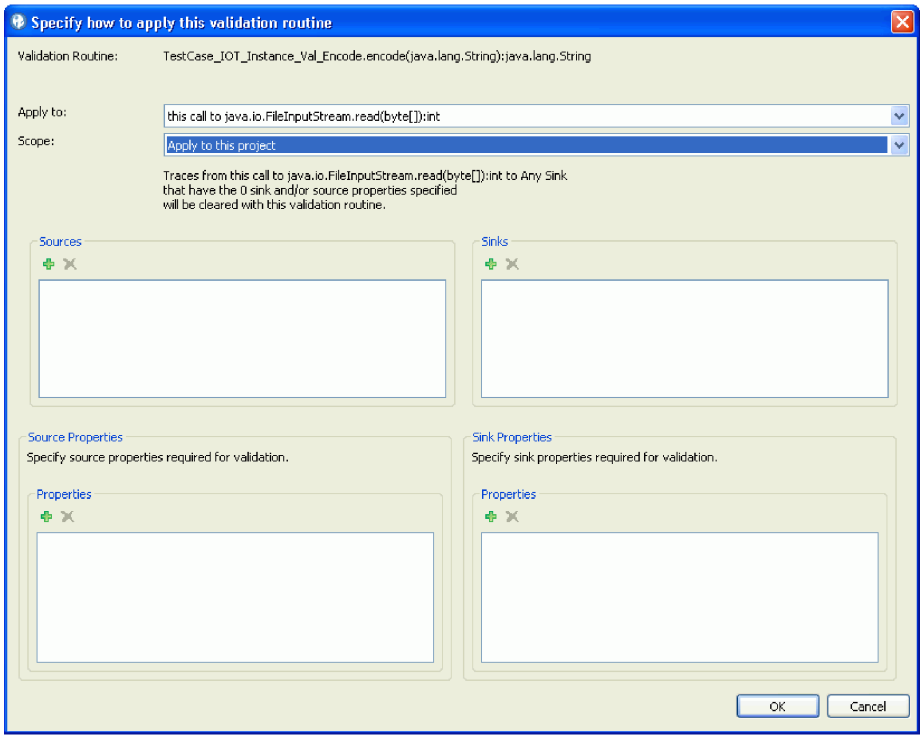
Procedure

1. In the Trace view, right-click the method or node for which you want to create a custom rule and then choose the custom rule to create - or select the method or node and click the appropriate custom rule toolbar button. The options for marking routines and methods are:

Option	Description
Mark as a Validation/Encoding routine	✓
Mark as not a Validation/Encoding routine	✓
Mark as a taint propagator	▣
Mark as not susceptible to taint	▣
Mark as a sink	■

   Note: If there is no entry in the Trace view for the method for which you want to create a custom rule, click **Launch the custom rules wizard to add a validation routine that is not on the trace graph**. In the Custom Rules Wizard, proceed to the Select Validation/Encoding Routine page. Select the validation routine and then specify the location, scope, any sources or sinks, or any properties, according to the instructions in the next step. See "Example 2: Creating a Validation/Encoding Routine from the Custom Rules Wizard" on page 153 for details about creating a validation routine with this wizard.

2. If you are creating a custom rule that marks a method as a sink or a validation/encoding routine, you may need to make further settings:
   a. If you mark the method as a sink, specify the sink attributes:
      - **Vulnerability Type**
      - **Severity**
   b. For validation routines, specify the location and scope - and any sources or sinks, or their properties, for which the validation routine should apply.



Validation Routine: TestCase_IOT_Instance_Val_Encode.encode(java.lang.String):java.lang.String

Apply to: this call to java.io.FileInputStream.read(byte[]):int

Scope: Apply to this project

Traces from this call to java.io.FileInputStream.read(byte[]):int to Any Sink
that have the 0 sink and/or source properties specified
will be cleared with this validation routine.

Specify source properties required for validation.

Specify sink properties required for validation.

- **Apply to**:
  - **this call to <method name>** (call site specific): Applies to the input just for this call.
  - **any call <method name>** (API specific): Applies to the validation/encoding routine for any call to the method.
  - **<method name> not considered, all constraints specified below**: Allows all sources to be affected by the rule.
- **Scope**:
  - **Apply to this project**: When selected, the rule is stored in the project (`.ppf`) file.
  - **Apply to all projects**: Validation rules created with this setting are stored in the database.
- **Sources**: Select the input source or sources to which the validation routine should apply. To add a source, click **Add** and then select the source from the Choose Signatures dialog box. To add multiple sources, you can multiselect them in the Choose Signatures dialog box.

- **Sinks**: Select the sink or sinks to which the validation routine should apply. To add a sink, click **Add** and then select the sink from the Choose Signatures dialog box. To add multiple sinks, you can multiselect them in the Choose Signatures dialog box.
- **Source Properties**: If you want the rule to clear traces that begin in a source with a specific property, click **Add a VMAT property** and then select the property from the Choose Properties dialog box. To add multiple properties, you can multiselect them in the Choose Properties dialog box.
- **Sink Properties**: If you want the rule to filter out traces that end in a sink with a specific property, click **Add a VMAT property** and then select the property from the Choose Properties dialog box. To add multiple properties, you can multiselect them in the Choose Properties dialog box.

3. After creating custom rules in the Trace view, you must scan your code again to see the rules reflected in the findings lists and traces. Custom rules that you create in the Trace view can be viewed and deleted in the Custom Rules view. To view details of the rule in the Custom Rules view, select the rule and click **Custom Rule Information**.

Code examples for tracing

This section provides code examples which illustrate tracking tainted data from a source to a sink - and how to create a validation and encoding routine.

- "Example 1: From source to sink"
- "Example 2: Modified from source to sink" on page 149
  - "Example 2: Creating a Validation/Encoding Routine from the Trace view" on page 150
  - "Example 2: Creating a Validation/Encoding Routine from the Custom Rules Wizard" on page 153
- "Example 3: Different source and sink files" on page 154
- "Example 4: Validation in depth" on page 155

Example 1: From source to sink

In the following code sample, the main method calls a method, `getVulnerableSource`, that returns a string. Note that, although the method reads data from a completely unknown file, it never checks the validity of the returned data. The main method then passes this tainted data into `writeToVulnerableSink`. The `writeToVulnerableSink` method writes the data out to the file, never checking its validity.

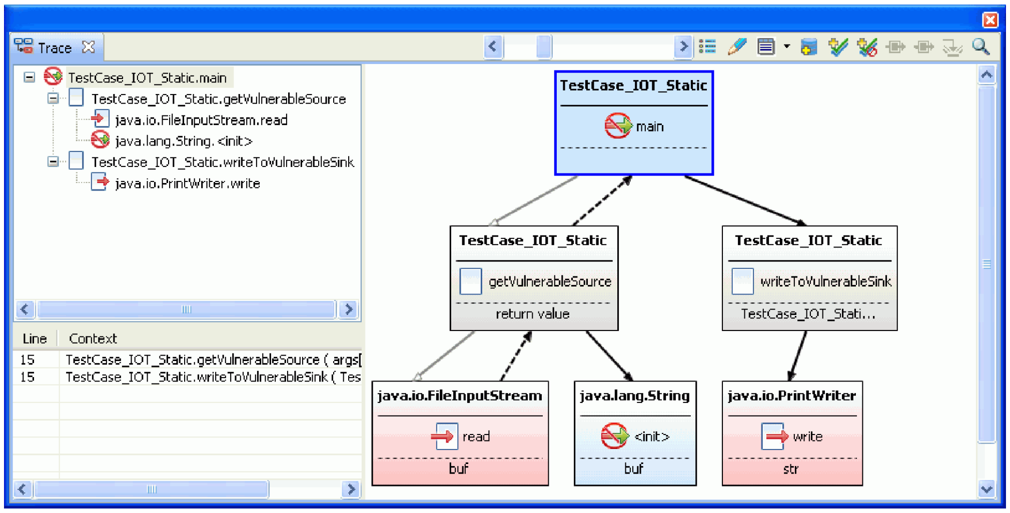
```
import java.io.*;

public class TestCase_IOT_Static {
  public static void main(String[] args) {
    try {
      writeToVulnerableSink(getVulnerableSource(args[0]));
    } catch (Exception e) {
    }
  }

  public static String getVulnerableSource(String file)
    throws java.io.IOException, java.io.FileNotFoundException {
    FileInputStream fis = new FileInputStream(file);
    byte[] buf = new byte[100];
    fis.read(buf);
    String ret = new String(buf);
    fis.close();
    return ret;
  }

  public static void writeToVulnerableSink(String str)
    throws java.io.FileNotFoundException {
    FileOutputStream fos = new FileOutputStream(str);
    PrintWriter writer = new PrintWriter(fos);
    writer.write(str);
  }
}
```

The code sample produces this trace:



```
TestCase_IOT_Static.main
  TestCase_IOT_Static.getVulnerableSource
    java.io.FileInputStream.read
    java.lang.String.<init>
  TestCase_IOT_Static.writeToVulnerableSink
    java.io.PrintWriter.write
```

Line	Context
15	TestCase_IOT_Static.getVulnerableSource (args[
15	TestCase_IOT_Static.writeToVulnerableSink (Tes

The pane shows the input stack where `main` calls `getVulnerableSource` which calls `FileInputStream.read` - and the output stack where `main` calls `writeToVulnerableSink`, which calls `PrintWriter.write`. The graph displays how the data flows from the read method to the write method, with `main` joining the two call stacks. The Data Flow section shows the line numbers in the operations in the `main` method that pass the taint. In this example, both method calls exist on the same line (line 15) within the method (in the sample code above, this translates to line number 7 - in the screen capture, the file includes 8 lines of commenting).

Example 2: Modified from source to sink

Example 2 is a modification of the Example 1 code. It enhances Example 1 by adding a validation routine, called `getVulnerableSource`, and an encoding routine called in `writeToVulnerableSink`.

```
import java.io.*;

public class TestCase_IOT_Instance_Val_Encode {
  public static void main(String[] args) {
    try {
      TestCase_IOT_Instance_Val_Encode testCase = new
        TestCase_IOT_Instance_Val_Encode();
      String file = args[0];
      String source = testCase.getVulnerableSource(file);
      source = testCase.validate(source);
      String encodedStr = testCase.encode(source);
      testCase.writeToVulnerableSink(file, encodedStr);
    } catch (Exception e) {
    }
  }

  public String getVulnerableSource(String file) throws Exception {
    FileInputStream fis = new FileInputStream(file);
    byte[] buf = new byte[100];
    fis.read(buf);
    fis.close();

    String ret = new String(buf);
    return ret;
  }

  public void writeToVulnerableSink(String file, String str)
      throws FileNotFoundException {
    FileOutputStream fos = new FileOutputStream(file);
    PrintWriter writer = new PrintWriter(fos);
    writer.write(str);
  }

  private String validate(String source) throws Exception {
    if (source.length() > 100) {
      throw new Exception("Length too long: " + source.length());
    }
  return source;
  }

  private String encode(String source) {
    return source.trim();
  }
}
```

The first scan produces a stack trace similar to the stack trace in Example 1.

Extending the Knowledgebase to include validation and encoding routines reduces noise in the findings and checks that validation and encoding routines are being called for all call graphs. For example, if you specified the data from any call to `java.io.FileInputStream.read(byte[]):int` in the previous example, the scan eliminates any calls from `read` that also called this validation routine. Also, calls from `read` that did not call the custom validation method are promoted to definitive security finding status, since not calling a known validation method in the code can lead to malicious attacks.

The validation routine may also validate the other variations of the `read` methods of the `FileInputStream`. These may be specified as additional sources. In addition, you may also know that only certain sinks (or sinks with certain properties) are validated by this method. For example, this routine could be restricted to sinks with the property of `Technology.IO`, such as the `PrintWriter.write` sink that is used to consume this example data.

Example 2: Creating a Validation/Encoding Routine from the Trace view

About this task

Since AppScan Source trace identifies the `FileInputStream.read` method as a source producing tainted data, you should create a validation or encoding routine to eliminate this finding from future scans.

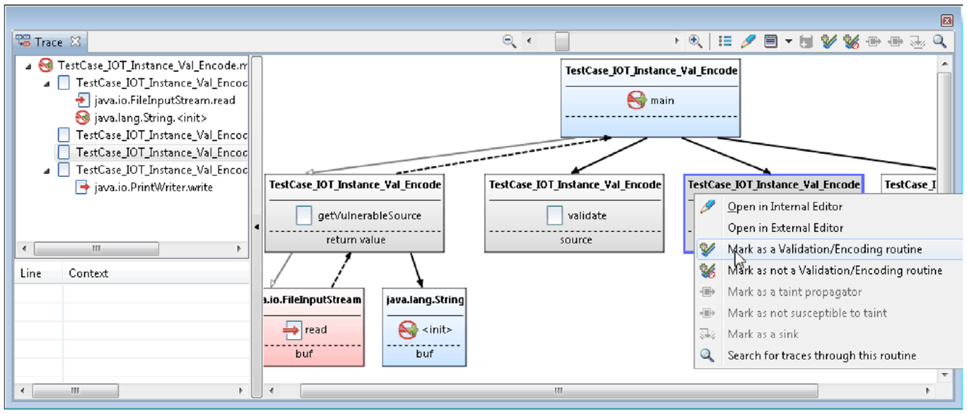
To create an input validation routine for `FileInputStream.read`:

Procedure

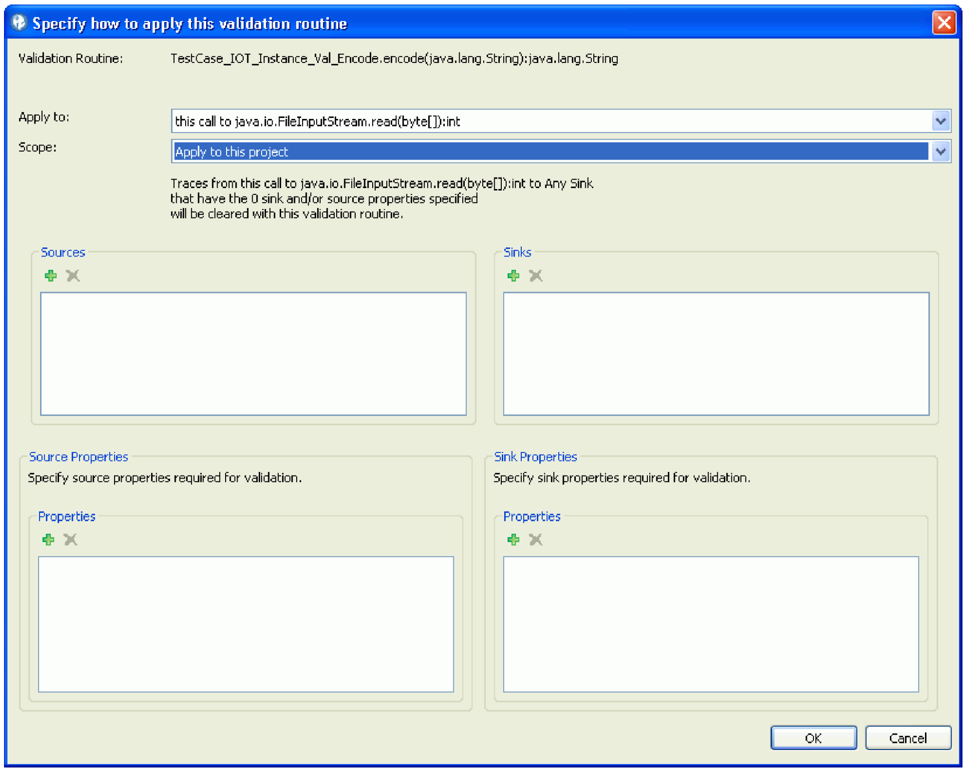
1. In the Trace view call graph, select and right-click the `TestCase_IOT_Instance_Val_Encode.encode` method.

Tip: If the validation/encoding routine that you want to create does not appear in the trace graph, you can create the routine by launching the Custom Rules Wizard from the Trace view. "Example 2: Creating a Validation/Encoding Routine from the Custom Rules Wizard" on page 153 explains the steps involved in doing this.

2. Select **Mark as a Validation/Encoding routine** in the menu.


Trace
TestCase_IOT_Instance_Val_Encode.m
TestCase_IOT_Instance_Val_Encoc
java.io.FileInputStream.read
java.lang.String.<init>
TestCase_IOT_Instance_Val_Encoc
TestCase_IOT_Instance_Val_Encoc
TestCase_IOT_Instance_Val_Encoc
java.io.PrintWriter.write
Line
Context
TestCase_IOT_Instance_Val_Encode
main
TestCase_IOT_Instance_Val_Encode
getVulnerableSource
return value
TestCase_IOT_Instance_Val_Encode
validate
source
TestCase_IOT_Instance_Val_Encode
TestCase_I
a.io.FileInputStream
read
buf
java.lang.String
<init>
buf
Open in Internal Editor
Open in External Editor
Mark as a Validation/Encoding routine
Mark as not a Validation/Encoding routine
Mark as a taint propagator
Mark as not susceptible to taint
Mark as a sink
Search for traces through this routine

3. If the `encode` routine only applies for this specific instance of calling `FileInputStream.read`, select **this call to java.io.FileInputStream.read** in the Specify how to apply this validation routine dialog box.



Specify how to apply this validation routine

Validation Routine: TestCase_IOT_Instance_Val_Encode.encode(java.lang.String):java.lang.String

Apply to: this call to java.io.FileInputStream.read(byte[]):int

Scope: Apply to this project

Traces from this call to java.io.FileInputStream.read(byte[]):int to Any Sink
that have the 0 sink and/or source properties specified
will be cleared with this validation routine.

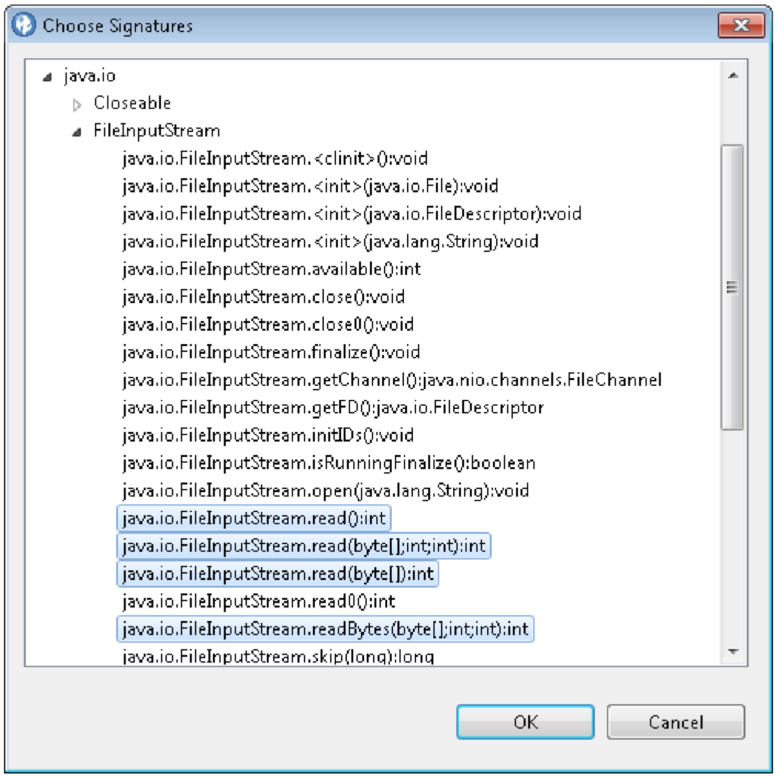
Source Properties
Specify source properties required for validation.

Sink Properties
Specify sink properties required for validation.

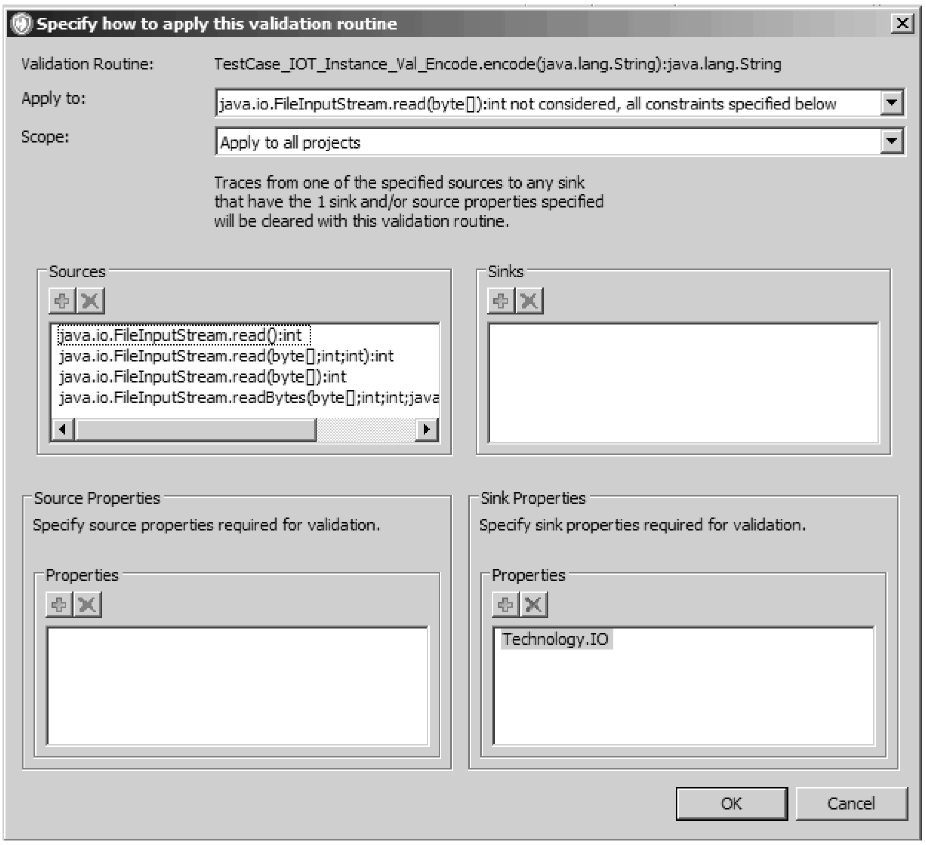
Typically, you would specify **this call to java.io.FileInputStream.read** because the `validate` method is private to the class and tightly associated with the code.

Select **any call to java.io.FileInputStream.read** to apply the validation routine for any call to the `read` method. When selecting this option, also select **Apply to this project** if this is only valid for the current project or **Apply to all projects**.

4. Set up the routine to apply to all `read` methods of the `FileInputStream` class and to any sink with a property of `Technology.IO` (such as the `java.io.PrintWrite.write` methods):
   a. **Adding the read methods as sources**: Although you could specify **any call to java.io.FileInputStream.read(byte[]):int** to add `java.io.FileInputStream.read(byte[]):int` as a source, we will instead add the sources individually. In the Specify how to apply this validation routine dialog box, select **java.io.FileInputStream.read(byte[]):int not considered, all constraints specified below** in the **Apply to** menu. Then click the **Sources** section **Add** button. In the Choose Signatures dialog box, expand the `java.io` and then `FileInputStream` sections. Multiselect the `java.io.FileInputStream.read*` nodes and then click **OK**.


Choose Signatures
java.io
Closeable
FileInputStream
java.io.FileInputStream.<clinit>():void
java.io.FileInputStream.<init>(java.io.File):void
java.io.FileInputStream.<init>(java.io.FileDescriptor):void
java.io.FileInputStream.<init>(java.lang.String):void
java.io.FileInputStream.available():int
java.io.FileInputStream.close():void
java.io.FileInputStream.close0():void
java.io.FileInputStream.finalize():void
java.io.FileInputStream.getChannel():java.nio.channels.FileChannel
java.io.FileInputStream.getFD():java.io.FileDescriptor
java.io.FileInputStream.initIDs():void
java.io.FileInputStream.isRunningFinalize():boolean
java.io.FileInputStream.open(java.lang.String):void
java.io.FileInputStream.read():int
java.io.FileInputStream.read(byte[];int;int):int
java.io.FileInputStream.read(byte[]):int
java.io.FileInputStream.read0():int
java.io.FileInputStream.readBytes(byte[];int;int):int
java.io.FileInputStream.skip(long):long
OK
Cancel

   b. **Adding the sink property**: Click the **Sink Properties** section **Add a VMAT property** button. In the Choose Properties dialog box, select the `Technology.IO` property and then click **OK**.
   c. When all settings are complete, the dialog box should look similar to this:


Specify how to apply this validation routine

Validation Routine: TestCase_IOT_Instance_Val_Encode.encode(java.lang.String):java.lang.String

Apply to: java.io.FileInputStream.read(byte[]):int not considered, all constraints specified below

Scope: Apply to all projects

Traces from one of the specified sources to any sink
that have the 1 sink and/or source properties specified
will be cleared with this validation routine.

Sources

java.io.FileInputStream.read():int
java.io.FileInputStream.read(byte[];int;int):int
java.io.FileInputStream.read(byte[]):int
java.io.FileInputStream.readBytes(byte[];int;int;java

Sinks

Source Properties

Specify source properties required for validation.

Properties

Sink Properties

Specify sink properties required for validation.

Properties

Technology.IO

5. Click **OK** to add the validation routine to the database.

Example 2: Creating a Validation/Encoding Routine from the Custom Rules Wizard

If the validation/encoding routine that you want to create does not appear in the trace graph, you can create the routine by launching the Custom Rules Wizard from the Trace view.

About this task

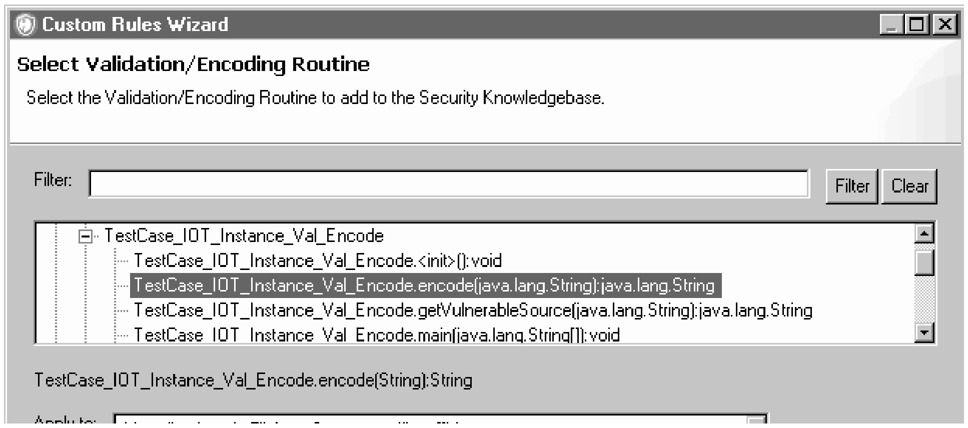
This example will create the same validation routine that is created in "Example 2: Creating a Validation/Encoding Routine from the Trace view" on page 150 - however, in this example, the routine will be created using the Custom Rules Wizard.

Procedure

1. In the Trace view, click **Launch the custom rules wizard to add a validation routine that is not on the trace graph** on the toolbar.

   Note: You cannot create a validation routine from the Custom Rules Wizard when it is launched from the Custom Rules view.
2. In the Select Validation/Encoding Routine page of the wizard, specify the location of the validation routine.

   For this example, select this routine:
   ```
   TestCase_IOT_Instance_Val_Encode.encode(java.lang.String):
   java.lang.String
   ```



Select Validation/Encoding Routine

Select the Validation/Encoding Routine to add to the Security Knowledgebase.

```
TestCase_IOT_Instance_Val_Encode
    TestCase_IOT_Instance_Val_Encode.<init>():void
    TestCase_IOT_Instance_Val_Encode.encode(java.lang.String):java.lang.String
    TestCase_IOT_Instance_Val_Encode.getVulnerableSource(java.lang.String):java.lang.String
    TestCase_IOT_Instance_Val_Encode.main(java.lang.String[]):void
```

TestCase_IOT_Instance_Val_Encode.encode(String):String

3. Complete the remaining sections of the wizard page with the same settings that were made in the **Specify how to apply this validation routine** dialog box in "Example 2: Creating a Validation/Encoding Routine from the Trace view" on page 150.
4. Click **Finish** to add the validation routine to the database.

Example 3: Different source and sink files

The following example illustrates the source in a different file from the sink.

TestCase_IOT_Xfile_Part1.java:

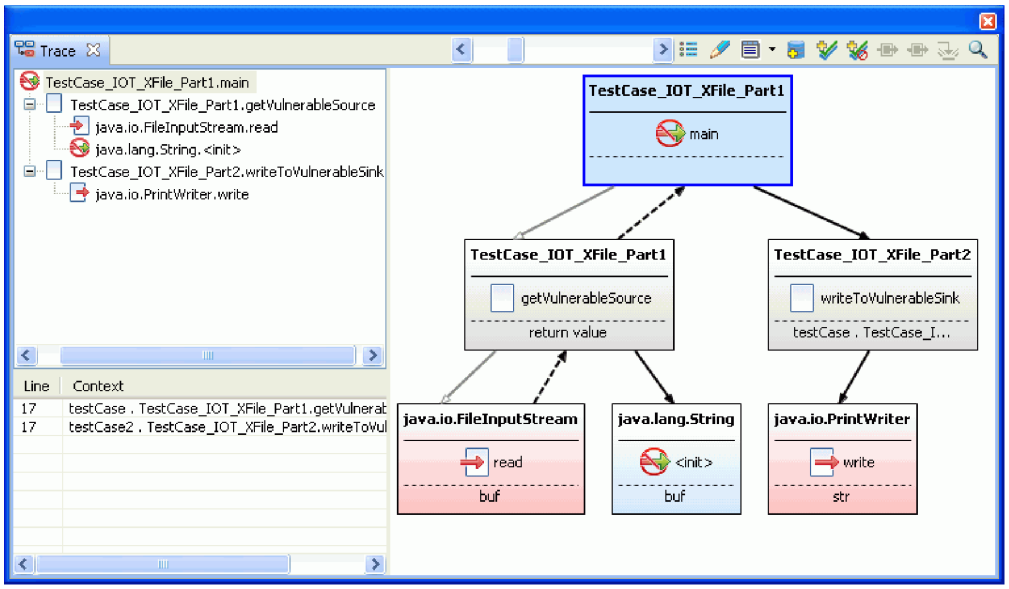
```
public class TestCase_IOT_XFile_Part1 {
 public static void main(String[] args) {
  try {
   TestCase_IOT_XFile_Part1 testCase =
    new TestCase_IOT_XFile_Part1();
   TestCase_IOT_XFile_Part2 testCase2 =
    new TestCase_IOT_XFile_Part2();
   testCase2.writeToVulnerableSink(
    testCase.getVulnerableSource(args[0]));
  } catch (Exception e) {
  }
 }

 public String getVulnerableSource(String file)
  throws IOException, FileNotFoundException {
  FileInputStream fis = new FileInputStream(file);
  byte[] buf = new byte[100];
  fis.read(buf);
  String ret = new String(buf);
  fis.close();
  return ret;
 }
}
```

TestCase_IOT_Xfile_Part2.java:

```
public class TestCase_IOT_XFile_Part2 {
 public void writeToVulnerableSink(String str)
  throws FileNotFoundException {
  FileOutputStream fos = new FileOutputStream(str);
  PrintWriter writer = new PrintWriter(fos);
  writer.write(str);
 }
}
```

Tracing the data from `TestCase_IOT_Xfile_Part1.java` to `TestCase_IOT_Xfile_Part2.java` allows data flow to be traced through an entire program. The stack trace appears:



```
TestCase_IOT_XFile_Part1.main
  TestCase_IOT_XFile_Part1.getVulnerableSource
    java.io.FileInputStream.read
    java.lang.String.<init>
  TestCase_IOT_XFile_Part2.writeToVulnerableSink
    java.io.PrintWriter.write
```

Line	Context
17	testCase . TestCase_IOT_XFile_Part1.getVulnerat
17	testCase2 . TestCase_IOT_XFile_Part2.writeToVul

This example shows the data flowing from `TestCase_IOT_XFile_Part1` to `TestCase_IOT_XFile_Part2` through the main method.

Example 4: Validation in depth

When you scan the Example 4 code, the first scan includes three AppScan Source traces with a root at the corresponding trace routines. Assume the selection of the `FileInputStream.read` method in `trace1` and the addition of the `validate` routine. The section following the sample source code describes the effects of each scope for the validation routine.

```
public class TestCase_IOT_UserValidation {
    ResultSet resultSet;
    FileInputStream fileInputStream;
    PrintWriter printWriter;
    byte[] buffer;

    public static void main(String[] args) throws Exception {
        TestCase_IOT_UserValidation testCase = new TestCase_IOT_UserValidation();
        testCase.trace1();

        TestCase_IOT_UserValidation testCase2 = new TestCase_IOT_UserValidation();
        testCase2.trace2();

        TestCase_IOT_UserValidation testCase3 = new TestCase_IOT_UserValidation();
        testCase3.trace3();
    }

    private void trace1() throws Exception {
        String source = getVulnerableSource1();
        source = validate(source);
        writeToVulnerableSink(source);
    }

    private void trace2() throws Exception {
        String source = getVulnerableSource2();
        source = validate(source);
        writeToVulnerableSink(source);
    }
    private void trace3() throws Exception {
        String source = getVulnerableSource3();
        source = validate(source);
        writeToVulnerableSink(source);
    }

    public String getVulnerableSource1() throws Exception {
        fileInputStream.read(buffer);
        return new String(buffer);
    }

    public String getVulnerableSource2() throws Exception {
        fileInputStream.read(buffer);
        return new String(buffer);
    }

    public String getVulnerableSource3() throws Exception {
        return resultSet.getString("x");
    }

    public void writeToVulnerableSink(String str) throws Exception {
        printWriter.write(str);
    }

    private String validate(String source) throws Exception {
        // validate
        return source;
    }
}
```

Call site specific validation routine - input for this call to `FileInputStream.read`

Create a call site specific validation routine when the validation only fits in a very narrow context or where the input method is too generic to supply one validation routine. When you **Apply to this call to FileInputStream.read** in the `trace1` method, `trace1` does not appear as a finding after the next scan because its call stack includes a call to the `validate` method. However, `trace2` is still reported even though it calls `validate`, because the scope of the validation routine is tied to the `trace1` call site. The `trace3` method also calls `validate`, but it continues to be reported because it uses `ResultSet.getString` as a source.

API specific validation routine - input for any call to `FileInputStream.read`

Create an API specific validation routine when the validation is only applicable for a particular source. When you **Apply to any call to FileInputStream.read** method, both the `trace1` and `trace2` methods are clear of findings on the next scan because they include a call to the `validate` method. However, the `trace3` method continues to exist even though it calls `validate` because it uses `ResultSet.getString` as a source.

Chapter 7. AppScan Source for Analysis and defect tracking

AppScan Source for Analysis integrates with defect tracking systems to deliver confirmed software vulnerabilities directly to the developer desktop. Defect submission to a defect tracking system contains a textual description of the bug and a file that contains only the findings submitted with the defect.

You can track your software vulnerability defects with AppScan Source for Analysis's integration with various defect tracking systems, including IBM Rational ClearQuest, IBM Rational Team Concert, HP Quality Center, and Microsoft Team Foundation Server.

Before you submit a finding to a defect tracking system or mail the defect to a developer, you may need to configure the defect tracking system preferences (see "Enabling defect tracking with preferences" on page 69).

Enabling defect tracking with preferences

Defect Tracking System preferences allow you to enable the submission of findings to a defect tracking system - and determine how defects are submitted.

The General tab in the Defect Tracking System preference page is used to enable or disable the Defect Tracking System integration feature in AppScan Source. If the **Enable Defect Tracking System Integration** checkbox is selected, the **Submit Defect** context menu action will be available for assessment findings. The General tab also provides discrete control over which Defect Tracking Systems will be available when submitting defects.

To learn about the preferences that can be set for supported defect tracking systems, refer to these help topics:
- "Rational ClearQuest preferences" on page 69
- "Quality Center preferences" on page 70
- "Rational Team Concert preferences" on page 72
- "Team Foundation Server preferences" on page 74

Rational ClearQuest preferences

To be able to complete Rational ClearQuest preferences, your Rational ClearQuest administrator must provide you with the required Rational ClearQuest settings. The settings are specific to your Rational ClearQuest environment.

Note: When integrating with Rational ClearQuest Version 8.0, the Rational ClearQuest schema must contain the fields that are available in the **DefectTracking** predefined schema.

Database set

A collection of one or more defect databases.

```
Linux default = Connection Name,
Windows default = Database Set
```

Database name

Name of the database to which to submit defects.

Database user name

Default Rational ClearQuest database user name.

Location of CQPerl executable

Location of the Rational ClearQuest CQPerl executable on the local computer. The provided default location maps to the default Rational ClearQuest installation location.

Entity for defect records

Entity (database object) configured by the Rational ClearQuest installation for use for defect objects.

The default entity is **Defect**.

Description field on record

Default Description is **Description**.

Headline field on record

Default Headline is **Headline**.

Single defect per finding

Submit a group of findings as a single defect or as multiple defects. You can change the submission method when you create the defect.

Quality Center preferences

You must first enable HP Quality Center as a General Defect Tracking System preference and then set the individual preference on the Quality Center tab.

Server URL

The Quality Center Server URL - for example, `http://<hostname>:<port>/qcbin/` or `https://<hostname>:<port>/qcbin/`.

User Name (Optional)

A user name to log in to Quality Center

Password (Optional)

If you entered a user name, enter the password for it

Domain

The Quality Center domain to which to connect.

Project

The Quality Center project to which to connect

Auto-login

When true, AppScan Source does not prompt for login information when submitting findings, and logs in with the default credentials specified in the Preferences. When false, you must log in each time you submit a finding to Quality Center.

Auto-submit

When true, the dialog box that is used for submitting new defects does not appear when submitting findings. AppScan Source for Analysis uses the **Default Defect Properties** specified in the Preferences. When false, a prompt appears requesting that you enter defect information (Severity, Priority, Defect Type, Status, and so forth) when submitting findings.

Resubmit previously-submitted findings

Findings submitted to Quality Center are tagged with Quality Center defect information (Defect ID, submitting user, and submission date). By default, AppScan Source does not resubmit the same finding more than once. This allows you to dispatch multiple findings to Quality Center, only entering new findings in the Quality Center database. When selected (true), previously submitted findings can be resubmitted to Quality Center.

Submit each finding as an individual bug

When submitting multiple findings in a single operation, you can either submit all findings as a single Quality Center defect or as a separate Quality Center defect for each individual AppScan Source finding. Selecting this check box sets the flag to true, creating a separate Quality Center defect for each individual finding. Setting the flag to false creates one Quality Center defect for all findings submitted as part of a bulk submission.

Auto Generate Bug Summary

When true, AppScan Source automatically generates a defect summary for the submission in Quality Center. The summary indicates the number of findings included in the defect and the type of findings included, such as `Validation.Required`.

When false, the Summary field appears for you to complete when submitting the defect in the dialog box that opens when you create a new defect.

Auto Load Bug Fields

Default setting is true. When the check box is selected, AppScan Source automatically loads defect field definitions from the Quality Center database, based on the current user and group settings in Quality Center. When false, AppScan Source does not display defect fields from Quality Center in the dialog box that opens when you create a new defect.

Default Defect Properties

To set default values for the different Quality Center defect attributes, click **Default Defect Properties** on the Quality Center preference tab.The default values either pre-populate the **New Defect** dialog box at submission time, or they are sent to Quality Center silently if the **Auto-submit** preference is selected.

Note: Defect properties and their available values are pulled dynamically from Quality Center each time the **Defect Properties** dialog box appears if **Auto Load Bug Fields** is selected. Therefore, any new fields and values added to the Quality Center database automatically appear in AppScan Source for Analysis. Valid server, login, and connection information is required to open and populate the **Defect Properties** dialog box with Quality Center information.

Customizing Quality Center defect fields

Through a configuration file, you can customize the fields and interactions between these fields in the New Defect dialog box. You can find an example configuration file in `<data_dir>\config\qc.dts` (where `<data_dir>` is the location of your AppScan Source program data, as described in "Installation and user data file locations" on page 258), which contains sample customizations and additional documentation. These customizations allow you to model your Quality Center Workflow script logic directly in the New Defect dialog box.

Available customizations include:
- Displaying custom fields, missing fields, or both
- Forcing fields to always display (overriding Quality Center settings)
- Updating required state of fields based on selection of other fields
- Dynamically updating list box options for a field based on the list box selection in another field

Rational Team Concert preferences

The Rational Team Concert preference tab allows you to configure a connection to a Rational Team Concert server and also to configure the values of work item attributes.

Once you have entered your connection information and successfully logged in, you can then choose to connect to one or more project areas. Each project area can have its own configuration of attribute preset values.

Note: When you connect to Rational Team Concert (by configuring preferences or submitting defects), you may be prompted to accept an SSL certificate. See "Rational Team Concert SSL certificates" on page 73 for more information.

To configure the attribute values for a given project area, select the project area and choose **Configure**. In the configuration dialog box, you can set attribute values to either hardcoded values or in some cases to variables that refer to a selected finding. For example, the use of `{Finding.fileName}` in an attribute value will be replaced with the actual source code file name for a finding during submission. Content Assist (`<Ctrl>+<Space>`) is provided for attribute values that support these variables. Teams are encouraged to share these configurations using the **Import** and **Export** buttons that are available on the main Rational Team Concert preference page.

Resolving Rational Team Concert server mismatches

AppScan Source can interact with only one Rational Team Concert server version at a time - either Version 2.x or Version 3.x.

Procedure

1. To toggle the Rational Team Concert version, perform one of these actions:
   - To enable Rational Team Concert Version 2.x, copy the `<install_dir>\configuration\rtc2config\config.ini` file up one level to `<install_dir>\configuration` (where `<install_dir>` is the location of your AppScan Source installation).
   - To enable Rational Team Concert Version 3.x, copy the `<install_dir>\configuration\rtc3config\config.ini` file up one level to `<install_dir>\configuration`.
2. Restart AppScan Source.

   Note: It may also be necessary to delete your `<user_home>\.ounceconfiguration` directory before restarting AppScan Source for this change to take effect (where `<user_home>` is your operating system home directory (for example, on Windows, the directory might be `C:\Documents and Settings\Administrator\`)).

Team Foundation Server preferences

The Team Foundation Server preference tab allows you to configure a connection to a Microsoft Team Foundation Server and to configure the values of work item fields.

Once you have entered your connection information and successfully logged in, you can then choose to connect to one or more projects.

Note: When configuring the login to Team Foundation Server 2010, the Server URL must contain the Team Project Collection you want to connect to. For example, `http://myserver:8080/tfs/`*`DefaultCollection`*.

Each project can have its own configuration of field preset values.

To configure the field values for a given project, select the project and choose **Configure**. In the configuration dialog box, you can set field values to either hardcoded values or in some cases to variables that refer to a selected finding. For example, the use of `{Finding.fileName}` in a field value will be replaced with the actual source code file name for a finding during submission. Content Assist (`<Ctrl>`+`<Space>`) is provided for fields that support these variables.

Teams are encouraged to share these configurations using the **Import** and **Export** buttons that are available on the main Team Foundation Server preference page.

Integrating HP Quality Center and AppScan Source for Analysis

The HP Quality Center integration with AppScan Source for Analysis requires a Quality Center client installation on the local computer. The Quality Center client application is downloaded and installed on the local computer the first time you log in to Quality Center through the Quality Center browser-based client interface.

Configuring Quality Center Information

Quality Center is configured in the Defect Tracking System preferences, on the Quality Center tab. You must enable Quality Center and set the Quality Center preferences before submitting AppScan Source findings as defects. See “Quality Center preferences” on page 70 for a description of each preference setting.

Note: In some environments (for example, those running HP Quality Center Version 11), you may need to install the HP ALM Client MSI Generator add-in for HP Quality Center integration to function.

Submitting findings to Quality Center

Findings are submitted to Quality Center through any AppScan Source for Analysis Findings view.

Procedure

1. Select the finding or findings in the table, or open a bundle. (If you open a bundle, select the bundle findings to submit.)
2. Right-click the selection and choose **Submit Defect** > **Dispatch to Quality Center** from the menu.
3. Log in to Quality Center.

   If your preference is configured to auto-login, the login dialog box does not appear. AppScan Source logs in with the default credentials.
4. Submit the findings.

   If your preference is configured to auto-submit, AppScan Source submits the finding information using the **Default Defect Properties** preferences.

Results

After the findings are submitted, an informational message appears indicating the number of successfully submitted findings.

Tracking findings submitted to Quality Center

Findings submitted to Quality Center are tagged with submission information:
- Quality Center Defect ID
- Submission date
- Quality Center user name

Submission information appears in the **Defect ID**, **Defect Date**, and **Defect User** columns in a findings table in any findings view. However, the default Findings Table does not contain these columns. You must configure the table to include these columns by clicking the **Select and Order Columns** toolbar button and customizing the table. See “Customizing the findings table” on page 243 for details about adding columns to the findings views.

Defect information persists from scan to scan, allowing you to track the status of an AppScan Source finding during triage and remediation.

AppScan Source finding information in Quality Center

When AppScan Source for Analysis creates a defect in the Quality Center database, the finding information is set as the defect **Description**. This finding information includes the Severity, Type, API, and Classification.

The Quality Center defect can also contain an AppScan Source bundle file (`.ozbdl`) that is added as an attachment to the defect. The bundle file contains all relevant information about the AppScan Source finding, including the trace. Developers can then save and open the bundle in AppScan Source for Analysis or Developer Plug-in and triage the defect.

Integrating Rational ClearQuest and AppScan Source for Analysis

The Rational ClearQuest integration with AppScan Source for Analysis requires a Rational ClearQuest client installation on the local computer. This installation includes the `CQPerl` executable, and you must configure the location of the executable in the AppScan Source for Analysis Rational ClearQuest preferences.

When you configure the Rational ClearQuest integration preferences, you specify information about the defect database schema. A Rational ClearQuest entity refers to Rational ClearQuest database objects, and you must specify the entity that your Rational ClearQuest installation uses for defects.

Note: The default location of the CQPerl executable that is needed for AppScan Source for Analysis integration is the default Rational ClearQuest installation directory.

Note: When integrating with Rational ClearQuest Version 8.0, the Rational ClearQuest schema must contain the fields that are available in the **DefectTracking** predefined schema.

Submitting findings to Rational ClearQuest

Findings integrate with your corporate defect tracking system for developer remediation. You can send individual findings to your defect tracking system - or you can submit bundles with one or more findings. The first time you submit a finding from AppScan Source to Rational ClearQuest during an AppScan Source session, you must log in with your user name and password.

When you submit a bundle to Rational ClearQuest, the bug number associates with the specific findings in the bundle, rather than the bundle itself. This ensures that you can manipulate the bundle further while preserving the specific findings associated with the defect at the time of defect creation.

A bundle can contain many findings. You have the option of submitting all findings as one defect or submitting each finding as a separate defect. If you select the **Single defect per finding** preference and multiple findings exist, you can edit the Description for those defects. You can only edit the Description of a single defect submission.

Note: You must set Default Tracking System preferences before you log in to Rational ClearQuest.

Note: When integrating with Rational ClearQuest Version 8.0, the Rational ClearQuest schema must contain the fields that are available in the **DefectTracking** predefined schema.

Submitting defects to Rational ClearQuest

Procedure

1. Select the finding or findings in the table, or open a bundle. (If you open a bundle, select the bundle findings to submit.)
2. Right-click the selection and choose **Submit Defect** > **Dispatch to ClearQuest** from the menu.
3. Log in to Rational ClearQuest and submit the findings.

Results

An assessment file that contains only the relevant files attaches to each defect. AppScan Source for Analysis or AppScan Source for Development can open this assessment file.

Note: When integrating with Rational ClearQuest Version 8.0, the Rational ClearQuest schema must contain the fields that are available in the **DefectTracking** predefined schema.

Integrating Rational Team Concert and AppScan Source for Analysis

Rational Team Concert integration with AppScan Source for Analysis does not require that an additional Rational Team Concert client be installed on your computer.

To configure a connection to Rational Team Concert, go to the Rational Team Concert tab in the Defect Tracking System preferences - or you can submit a defect and you will be prompted at that point to log in and configure your connection.

Rational Team Concert preferences also allow you to configure the preset field values that will be used during defect submission. This lets you set values that you want to use for every defect and also to modify the default values that ship with AppScan Source.

Note: When you connect to Rational Team Concert (by configuring preferences or submitting defects), you may be prompted to accept an SSL certificate. See "Rational Team Concert SSL certificates" on page 73 for more information.

Submitting defects to Rational Team Concert

You can submit bundles with one or more findings to Rational Team Concert - or you can submit individual findings. The first time you submit a finding from AppScan Source for Analysis to Rational Team Concert, you must log in with your username and password. If you want to configure the preset field values that will be used during submission, you can do so in the Rational Team Concert preferences.

About this task

When you submit a bundle to Rational Team Concert, the work item number is associated with the specific findings in the bundle, rather than the bundle itself.

This ensures that you can manipulate the bundle further while preserving the association of specific findings to work item numbers.

Procedure

1. Select the finding or findings in the table, or open the bundle. (If you open the bundle, select the bundle findings to submit.)
2. Right-click the selection and choose **Submit Defect** > **Dispatch to Rational Team Concert** from the menu.
3. The submission dialog box will then guide you through the process, including login, if necessary, and filling in required attributes.

   Note: When you connect to Rational Team Concert (by configuring preferences or submitting defects), you may be prompted to accept an SSL certificate. See "Rational Team Concert SSL certificates" on page 73 for more information.

Results

A bundle will be automatically added to the submitted work item that can then be opened at a later time by a user of AppScan Source for Analysis or AppScan Source for Development.

Resolving Rational Team Concert server mismatches

AppScan Source can interact with only one Rational Team Concert server version at a time - either Version 2.x or Version 3.x.

Procedure

1. To toggle the Rational Team Concert version, perform one of these actions:
   - To enable Rational Team Concert Version 2.x, copy the `<install_dir>\configuration\rtc2config\config.ini` file up one level to `<install_dir>\configuration` (where `<install_dir>` is the location of your AppScan Source installation).
   - To enable Rational Team Concert Version 3.x, copy the `<install_dir>\configuration\rtc3config\config.ini` file up one level to `<install_dir>\configuration`.
2. Restart AppScan Source.

   Note: It may also be necessary to delete your `<user_home>\.ounceconfiguration` directory before restarting AppScan Source for this change to take effect (where `<user_home>` is your operating system home directory (for example, on Windows, the directory might be `C:\Documents and Settings\Administrator\`)).

Rational Team Concert SSL certificates

When a Rational Team Concert server is installed, it should be configured to use a valid SSL certificate. If this is not done, you will receive an untrusted connection message when logging in to the server (while configuring preferences or submitting defects). This topic outlines Rational Team Concert SSL certificate considerations.

SSL certificate storage location

Certificates that have been permanently accepted are stored in `<user_home>/.jazzcerts` (where `<user_home>` is your operating system home directory (for example, on Windows, the directory might be `C:\Documents and Settings\Administrator\`). Removing `<user_home>/.jazzcerts` deletes all stored certificates for AppScan Source and Rational Team Concert clients.

SSL certificate sharing with Rational Team Concert clients

AppScan Source shares its certificate store with Rational Team Concert clients. If you permanently accept a certificate using a Rational Team Concert client, it will be reused by AppScan Source (you will not be prompted in AppScan Source to accept a certificate). Similarly, if you permanently accept a certificate in AppScan Source, it will be reused by Rational Team Concert clients.

Integrating Microsoft Team Foundation Server and AppScan Source for Analysis

The Team Foundation Server integration with AppScan Source for Analysis requires a Microsoft Visual Studio Team Explorer client installation on the local computer.

To configure a connection to your Team Foundation Server, go to the Team Foundation Server tab in the Defect Tracking System preferences - or you can submit a defect and you will be prompted at that point to log in and configure your connection.

The Team Foundation Server preferences also allow you to configure the preset field values that will be used during defect submission. This lets you set values that you want to use for every defect and also to modify the default values that ship with AppScan Source.

Submitting defects to Microsoft Team Foundation Server

You can submit bundles with one or more findings to Team Foundation Server - or you can submit individual findings. The first time you submit a finding from AppScan Source for Analysis to Team Foundation Server, you must log in with your username and password. If you want to configure the preset field values that will be used during submission, you can do so in the Team Foundation Server preferences.

About this task

When you submit a bundle to Team Foundation Server, the work item number is associated with the specific findings in the bundle, rather than the bundle itself. This ensures that you can manipulate the bundle further while preserving the association of specific findings to work item numbers.

Note: When configuring the login to Team Foundation Server 2010, the Server URL must contain the Team Project Collection you want to connect to. For example, `http://myserver:8080/tfs/`*`DefaultCollection`*.

Procedure

1. Select the finding or findings in the table, or open the bundle. (If you open the bundle, select the bundle findings to submit.)
2. Right-click the selection and choose **Submit Defect** > **Dispatch to Team Foundation Server** from the menu.

3. The submission dialog box will then guide you through the process, including login, if necessary, and filling in required fields.

Results

A bundle will be automatically added to the submitted work item that can then be opened at a later time by a user of AppScan Source for Analysis or AppScan Source for Development.

Working with submitted defects

When you submit more than a few findings as separate defects, the process runs in the background while you continue the triage process. After the defect submission, a defect ID received from the defect system is attached to the relevant findings and remains with that finding. To work with a defect that has been submitted to your defect tracking system, follow the steps in this topic.

Procedure

1. Open your defect tracking system and locate the defect.
2. Save the attachment as an AppScan Source bundle (`.ozbdl`) file. You can open this file in AppScan Source for Analysis or as a bundle in AppScan Source for Development.

Submitting bundles to defect tracking and by email

The findings in bundles can be submitted to your corporate defect tracking system - or sent by email. Once you place findings in a bundle, you can submit these findings as bugs for developer remediation.

Procedure

1. Open the bundle.
2. Click the **Submit bundle to defect tracking** toolbar button down arrow and then select your defect tracking system.

   Note: Depending on your defect tracking system, you may want to modify Defect Tracking System preferences before submitting the bundle.

   Alternatively, on the Bundle toolbar, click **Email Bundle** to send the bundle to others (email preferences must be configured beforehand).
3. Complete the configuration dialog boxes that open. These vary depending on the defect tracking system that you have chosen - and are described in the *AppScan Source for Analysis and defect tracking section* of the help.

Tracking defects through email (sending findings by email)

About this task

If you have configured email preferences, you can email findings or bundles directly to developers to advise them of potential defects found after a scan. The email includes an attachment that contains the findings - and text that describes the findings.

Note: Some Simple Mail Transfer Protocol (SMTP) relays only deliver mail to specific domains. In this case, if you send from `mydomain.com`, only recipients in `mydomain.com` can receive the email through AppScan Source for Analysis.

To email findings from a findings table:

Procedure

1. Select the finding or findings in the table, or open a bundle. If you open a bundle, select the bundle findings to mail.
2. Right-click the selection and choose **Email Findings** from the menu.
3. The email will include a bundle attachment that contains the findings. In the Attachment File Name dialog box, specify a name for the finding bundle. For example, specifying `my_finding` in the **Attachment File Name** field causes a bundle with file name `my_finding.ozbdl` to be attached to the email. Click **OK** to open the Email Findings dialog box.
4. By default, the **Mail To** field in the Email Findings dialog box will populate with the **To Address** that is specified in the email preferences - however, it can easily be changed when preparing the email. In this dialog box, review the contents of the email and then click **OK** to send the email.

Results

Example email contents:

```
1 findings:
Name: JavaAny.test_DataInput
Type: Vulnerability.Validation.Required
Severity: Low
Classification: Suspect
File Name: C:\TestApps\java\JavaAny\src\JavaAny.java
Line / Col: 275 / 0
Context: di . java.io.DataInput.readFully ( ba )
Notes: Check into this vulnerability and report back ASAP.
```

Tip: You can email individual findings or bundles from the Finding Detail view. You can also email bundles by clicking **Email Bundle** on the Bundle toolbar.

Chapter 8. Findings reports and audit reports

Security analysts and risk managers can access reports of select findings or a series of audit reports that measure compliance with software security best practices and regulatory requirements. This section explains how to create reports of aggregate finding data.

AppScan Source for Analysis generates two report types - Findings Reports and AppScan Source Reports. A *Findings Report* is a report of selected findings. An *AppScan Source Report* is a report based on categorized groupings of all findings tailored to a specific security policy. AppScan Source reports are listed in “AppScan Source reports” on page 171.

Reports provide the details about findings gathered during a particular scan, and all AppScan Source reports can contain any notes and trace data added to the findings. The length of the report depends on the number of findings included in the report. You can generate reports as PDF files or in Hypertext Markup Language (HTML). HTML reports function like web pages where you can jump to a section by clicking a button or link. Then you can navigate through the information using browse functions found in web browsers.

Reports also list any scan-time filters that have been applied to the findings. Scan-time filters are described in “Determining applied filters” on page 119.

Creating findings reports

About this task

After you scan, you may want to generate reports about the identified vulnerabilities. You can generate multiple findings reports:
- Findings
- Findings by Type
- Findings by Classification
- Findings by File
- Findings by API
- Findings by Bundle
- Findings by CWE (Common Weakness Enumeration)
- DTS Activity

Note: Findings reports show detailed findings by category, similar to the results in the findings table. The generation of findings reports can be memory-intensive and may require up to 1024 MB of additional system memory.

CWE ID hyperlinks in the findings report connect to the CWE website at http://cwe.mitre.org/.

To generate a findings report:

Procedure

1. In a view that contains findings, select the findings to include in the report. If you do not select any findings, the report consists of all findings in the active view.

   On the **Tools** menu, click **Generate Findings Report**. Alternatively, in views that contain findings, select and right-click a set of findings, and then select **Generate Findings Report** in the menu.
2. In the **Select Findings Report** dialog box, select a report type.

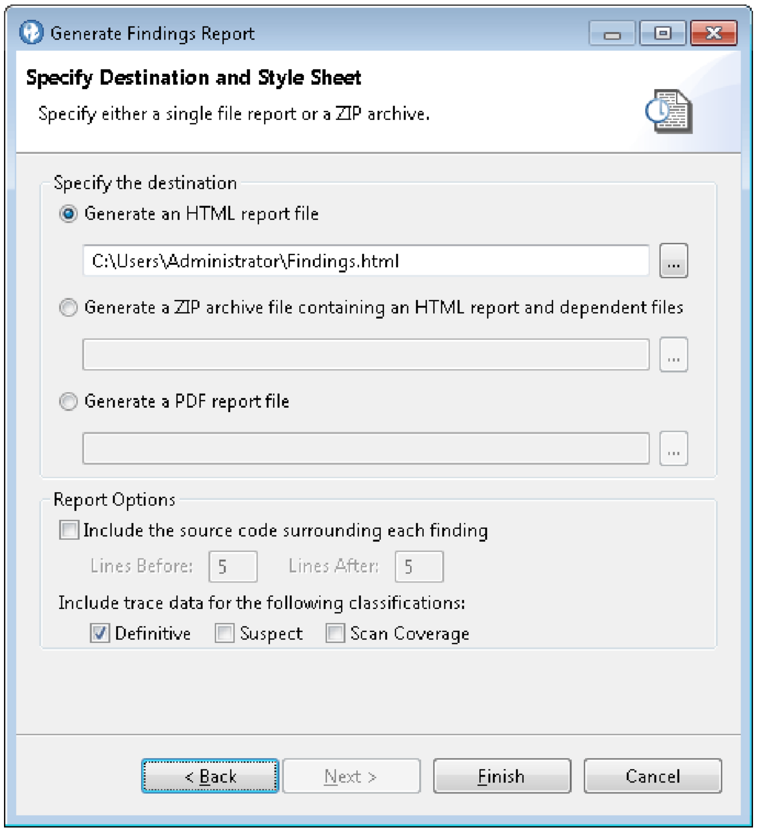
   Click **Finish** to generate the report - or click **Next** to specify these optional settings in the Specify Destination and Style Sheet page:
   - You can specify the report destination and format. You can generate the report in HTML format, as a ZIP file that contains all HTML report components, or a PDF (you must have Adobe Acrobat Reader to view PDF reports). If you do not specify a report destination and format (or click **Finish** in the Select Findings Report page), HTML is chosen by default, and the report is saved to `<data_dir>\reports` (where `<data_dir>` is the location of your AppScan Source program data, as described in "Installation and user data file locations" on page 258).

     Note: If you are creating a custom report (rather than a findings report) in PDF format, you can specify the level of detail to include in the report:
     - **Summary**: Contains counts for each report group
     - **Detailed**: Contains counts for each API for each vulnerability property
     - **Comprehensive**: Contains tables consisting of every finding for every API
     - **Annotated**: Contains all findings and any notes, trace data, or code snippets included with the findings
   - To include a code snippet in the report, select **Include the source code surrounding each finding** and indicate the number of lines before and after the vulnerable line of code to include in the report.

     Tip: In the Reporting section of the Finding Detail view, you can also set the number of lines of code to include before and after the finding in reports.

     After the report is generated, when you expand a finding that contains notes or code snippets, the source code appears below the finding in a blue box or below the yellow note. Bold red text highlights the vulnerable line of code.
   - To include AppScan Source trace data in the report, select one or more of the classifications (**Definitive**, **Suspect**, or **Scan Coverage**) under **Include trace data for the following classifications**.

   Click **Finish** to generate the report.


Generate Findings Report

Specify Destination and Style Sheet

Specify either a single file report or a ZIP archive.

Specify the destination

[● Generate an HTML report file]

C:\Users\Administrator\Findings.html

○ Generate a ZIP archive file containing an HTML report and dependent files

○ Generate a PDF report file

Report Options

☐ Include the source code surrounding each finding

Lines Before: 5 Lines After: 5

Include trace data for the following classifications:

☑ Definitive ☐ Suspect ☐ Scan Coverage

AppScan Source reports

AppScan Source reports help software security analysts, development managers, and risk management auditors measure compliance with software security best practices and regulatory requirements. AppScan Source reports help ensure that your critical applications meet the security standards you set.

AppScan Source uses source code vulnerability analysis results to power a series of reports that provide a detailed picture of compliance to a security, development, or audit professional.

AppScan Source reports feature:

- Report Card: Report card for a brief view of the security state of each major category
- Detailed Audit Review: A detailed audit of non-compliant findings
- Drill Down: Direct access to the non-compliant code for further analysis and prioritization of remediation and assignment

AppScan Source for Analysis generates a variety of AppScan Source reports:

- "CWE/SANS Top 25 2011 report" on page 173
- "DISA Application Security and Development STIG V3R6 report" on page 173
- "Open Web Application Security Project (OWASP) Mobile Top 10 report" on page 174
- "Open Web Application Security Project (OWASP) Top 10 2013 report" on page 173

- "Payment Card Industry Data Security Standard (PCI DSS) Version 1.1 and Version 2.0 reports" on page 174
- "Software Security Profile report" on page 174: Provides an overall view of the security state of an application, across every major vulnerability category.

Creating an AppScan Source custom report

Procedure

1. On the **Tools** menu, click **Generate Report**.
2. In the Generate Report dialog box, select an AppScan Source report:
   - **CWE SANS Top 25 2011**
   - **DISA Application Security and Development STIG V3R6**
   - **OWASP Mobile Top 10**
   - **OWASP Top 10 2013**
   - **PCI Data Security Standard V1.1**
   - **PCI Data Security Standard V2.0**
   - **Software Security Profile**

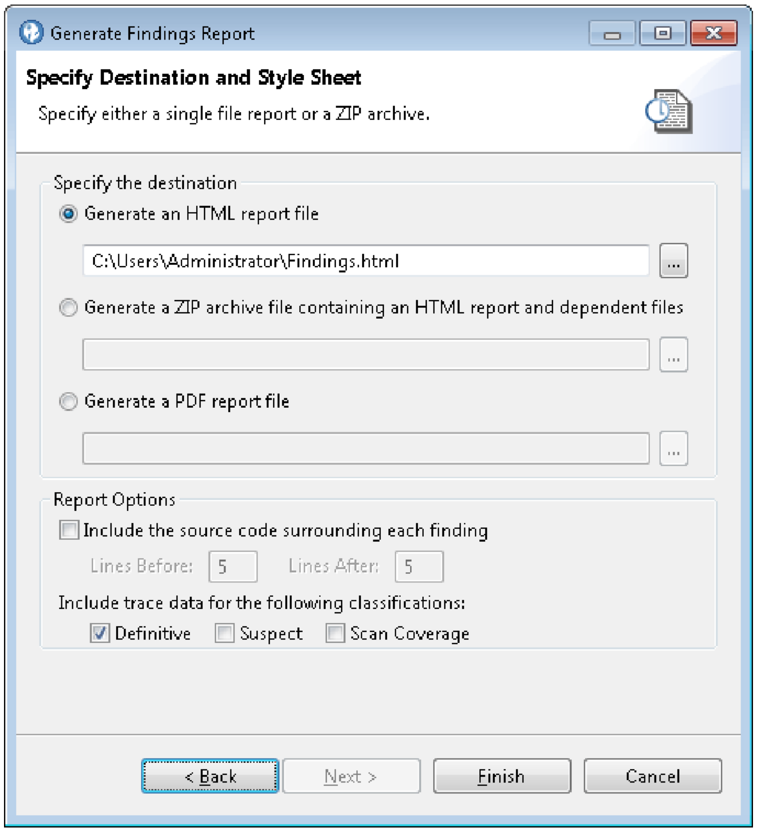
   Click **Finish** to generate the report - or click **Next** to specify these optional settings in the Specify Destination and Style Sheet page:
   - You can specify the report destination and format. You can generate the report in HTML format, as a ZIP file that contains all HTML report components, or a PDF (you must have Adobe Acrobat Reader to view PDF reports). If you do not specify a report destination and format (or click **Finish** in the Select Findings Report page), HTML is chosen by default, and the report is saved to `<data_dir>\reports` (where `<data_dir>` is the location of your AppScan Source program data, as described in "Installation and user data file locations" on page 258).

     Note: If you are creating a custom report (rather than a findings report) in PDF format, you can specify the level of detail to include in the report:
     - **Summary**: Contains counts for each report group
     - **Detailed**: Contains counts for each API for each vulnerability property
     - **Comprehensive**: Contains tables consisting of every finding for every API
     - **Annotated**: Contains all findings and any notes, trace data, or code snippets included with the findings
   - To include a code snippet in the report, select **Include the source code surrounding each finding** and indicate the number of lines before and after the vulnerable line of code to include in the report.

     Tip: In the Reporting section of the Finding Detail view, you can also set the number of lines of code to include before and after the finding in reports.

     After the report is generated, when you expand a finding that contains notes or code snippets, the source code appears below the finding in a blue box or below the yellow note. Bold red text highlights the vulnerable line of code.
   - To include AppScan Source trace data in the report, select one or more of the classifications (**Definitive**, **Suspect**, or **Scan Coverage**) under **Include trace data for the following classifications**.

   Click **Finish** to generate the report.


Generate Findings Report

Specify Destination and Style Sheet

Specify either a single file report or a ZIP archive.

Specify the destination

(●) Generate an HTML report file

C:\Users\Administrator\Findings.html

( ) Generate a ZIP archive file containing an HTML report and dependent files

( ) Generate a PDF report file

Report Options

[ ] Include the source code surrounding each finding

Lines Before: 5 Lines After: 5

Include trace data for the following classifications:

[✓] Definitive [ ] Suspect [ ] Scan Coverage

CWE/SANS Top 25 2011 report

The CWE/SANS Top 25 2011 report is based on the *2011 CWE/SANS Top 25 Most Dangerous Software Errors*.

To learn about the *2011 CWE/SANS Top 25 Most Dangerous Software Errors*, see http://cwe.mitre.org/top25/.

DISA Application Security and Development STIG V3R6 report

This topic provides links to the Defense Information Systems Agency (DISA) Application Security and Development Security Technical Implementation Guide (STIG) website and guidance documents.

To learn about the DISA Application Security and Development STIG, see http://iase.disa.mil/stigs/index.html. Guidance documents are available at http://iase.disa.mil/stigs/app_security/app_sec/app_sec.html.

Open Web Application Security Project (OWASP) Top 10 2013 report

This topic provides links to the Open Web Application Security Project (OWASP) website and guidance documents.

To learn about OWASP, see https://www.owasp.org/index.php/Main_Page. Links to various OWASP documents and security risks are available at https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project.

Open Web Application Security Project (OWASP) Mobile Top 10 report

This topic provides links to the Open Web Application Security Project (OWASP) website and guidance documents.

To learn about the OWASP Mobile Security Project, see https://www.owasp.org/index.php/OWASP_Mobile_Security_Project.

Payment Card Industry Data Security Standard (PCI DSS) Version 1.1 and Version 2.0 reports

This report provides relevant data needed to ensure compliance with the Payment Card Industry Data Security Standard (PCI DSS).

See https://www.pcisecuritystandards.org/security_standards/index.php for information.

Software Security Profile report

The Software Security Profile presents a comprehensive analysis of the characteristics of your application that have direct relevance to its security. It provides a detailed audit of critical security features in software for a particular project. This report helps you verify the implementation of requirements such as encryption, access control, logging, and error handling before certifying the software for deployment.

The composite identifies areas of potential risk and presents recommendations for minimizing those risks. The report helps facilitate an assessment of the overall application security - which is useful for compliance, policy, and architectural reviews. Findings are based on extensive static analysis of source code using a database of flaws, vulnerabilities, industry-specific standards, and general best practices.

The Software Security Profile displays this information:

- Report Card: Contains links to the report details and severity indicators summarizing the section.
- Overview: Summarizes the purpose of the report and describes the application configuration.
- Metrics: Identifies the total number of packages, classes, methods, and lines of code in all of the packages in the project.
- Detailed Findings by Category: Reports each vulnerability category found with a vulnerability category name and an icon that indicates the severity level of the vulnerability.

Chapter 9. Creating custom reports

In the Report Editor, you create report templates used to generate custom reports.

An AppScan Source Findings Report or AppScan Source report may not provide the exact data that you need; you may require that your report contains more or less information. The AppScan Source for Analysis Report Editor allows you to create custom reports.

Typically, you create a custom report when you must:

- Produce a report that maps to and reports on a unique security policy. You first create a custom report, and then apply the report to a specific assessment.
- Define and produce a report to highlight unique findings and characteristics.
- Modify or add to an existing report.

When you save the report template to `<data_dir>\reports` (where `<data_dir>` is the location of your AppScan Source program data, as described in "Installation and user data file locations" on page 258), the report is available for assessments of any application. When saved to the directory of a particular application, the report is available for scans of that application or any of its projects.

Before you begin creating or editing an AppScan Source report, familiarize yourself with the report types and the elements that comprise each report. When you create a custom report, you can map report elements in any order. Report elements include finding information, code snippets, traces, and remediation content, as well as text and graphical elements.

Report Editor

With the Report Editor, you can edit custom reports or templates or create a new report. Custom reports include any items that are available to a findings report, such as finding information, code snippets, AppScan Source trace, and remediation content, as well as a vulnerability matrix. Before you start designing new reports, it is recommended that you become familiar with the report creation process by modifying an existing report template in the Report Editor.

The Report Editor consists of the Report Layout, Categories, and Preview tabs.

- **Report Layout**: Design the appearance of the report. In the layout, you add, remove, and reorder AppScan Source report elements.
- **Categories**: Create and edit categories. A *category* is a group of findings. The category identifies the findings to include in the report, how to group those findings, and the grouping order.
- **Preview**: Look at the report for the current assessment as you edit it.

The three tabs contain common fields:

- **File**: Path of the saved grouping file (read-only). Nothing displays in this field until the file has been saved. Once saved, the grouping file is an XML file that defines the report.
- **Name**: User-defined report name.

Toolbar buttons for saving, opening, creating, copying, and generating custom reports include:

- **Create a new report**: Creates a new custom report
- **New report from existing**: Creates a new custom report from an existing report template
- **Open a saved report**: Open a grouping file to edit
- **Save**: Save the current report to the specified file
- **Save As**: Save the current report to a new file
- **Generate an instance of this report**: Create a copy of the report for the currently open assessment

Tip: To view samples of existing reports, click **New report from existing** and then choose one of the AppScan Source report templates. Exploring the Report Layout and Categories tabs in the templates will give you a feel for how reports are designed.

Report Layout tab

The Report Layout Tab consists of Palette and Layout sections - and sections that allow you to specify a header or footer that appears on each page.

Page Header and Page Footer

The **Page Header** field allows you to specify text that appears at the top of every report page - while the **Page Footer** field allows you to specify text that appears at the bottom of every page.

Palette

The Palette displays a list of elements that comprise AppScan Source standard reports. Some of the elements only display information for categories that have been defined in the Categories tab (see Table 19 on page 177).

Table 18. Report Layout Palette - elements that do not depend on categories

Report Element	**Description**
Text Header	Adds a bold block of text to the report layout.
Image Header	Displays an image scaled to a specified size in pixels.
AppScan Source Header	Report header containing AppScan Source branding.
Title and Date	Report title that includes name of the item that was scanned - along with the date of scan and the date the report was generated.
Text Block	Any user-defined text. A heading can also be added for the text block in the **Label** field.
Vulnerability Matrix	Assessment vulnerability matrix (displays the same graph that appears in the Vulnerability Matrix view).
Metrics	Identifies the total number of packages, classes, methods, and lines of code in all of the packages in the project.

Table 18. Report Layout Palette - elements that do not depend on categories (continued)

Report Element	Description
Scan History	Metrics for the current scan and historical metrics for scans of the same target.

Table 19. Report Layout Palette - elements that depend on categories

Report Element	Description
Report Card	Brief breakdown of vulnerability levels of every category defined in the Categories tab. Contains links to the report details and severity indicators summarizing the section.
Vulnerability Breakdown	Table with a breakdown of the number of vulnerabilities in all categories defined in the Categories tab, by severity and classification.
Partial Report Card	Breakdown of vulnerability levels of user-specified categories as specified in the Categories tab.
Categories	Lists all categorized findings data as defined in the Categories tab.
Category	Lists all findings in one or more categories that have been defined in the Categories tab.

Layout

As you add items from the palette, they appear in the Layout. Use the section toolbar to remove, modify, or move items in the layout.

Categories tab

Using the Categories tab, you can add categories that contain findings based on bundles, properties, or selected findings that you choose. The categories can then be used when adding certain items to the Layout. For example, when you add a Vulnerability Breakdown to the Layout, a table with a breakdown of the number of vulnerabilities in all categories (by severity and classification) is added to the layout. The Categories tab consists of a pane with a tree of categories and a pane in which to edit the attributes of the selected category. Each category contains the findings in the assessment that satisfy certain requirements that you define.

The available categories include:

- Bundle: A bundle category consists of a list of bundle names. Any finding in a bundle whose name appears in the list appears in this category. Although you choose bundles from the current assessment, you can apply the bundle category to any assessment since bundles are matched by name.
- Individual findings: Choose specific findings to add to the category. Only a snapshot of the finding is added to the report. If you modify the finding after it is added to the report, the report does not reflect the change.
- Vulnerability Types, Mechanisms, and Technologies properties: Choose sets of properties and required properties from APIs in the AppScan Source Security Knowledgebase. If a finding contains at least one of the **Properties** and all **Required Properties**, it is included in the report.

This table identifies the category panes and the items comprising the pane.

Table 20. Categories tab attributes

Attribute	**Description**	**How to edit**
Label	The brief name of the category, such as Buffer Overflow. The label identifies the category in the tree list of categories - and it is the category heading in the custom report.	Type a label in a single line text field.
Summary	A template for a sentence stating how many findings are reported in this category. The actual count replaces `%FindingCount%` during report generation.	Type a short description of the category and click **Add Count** to place the variable, `%FindingCount%` in the phrase at the cursor location.
Text	Brief category description.	Enter text describing the category.
Properties (Property categories only)	Findings having at least one of these properties will be reported in this category. If a finding does not have all listed required properties, then the finding is not included in this category.	Click **Add** on the toolbar and select a property from the Add Properties dialog box. Click **Remove** to remove the selected items from the list.
Required Properties (Property categories only)	Findings with all required properties and at least one property appear in the report under this category.	Click **Add** on the toolbar and select a property from the Add Properties dialog box. Click **Remove** to remove the selected items from the list.
Bundles (Bundle categories only)	Specifies the names of bundles to include in this category.	Click **Add a bundle** in the Bundles section, and select the bundles from the list.
Findings (Findings categories only)	Specifies the findings to include in this category.	Select findings in any findings table and then click **Add findings** on the table toolbar to add the selected findings. If more than one view contains selected findings, you will be prompted to select the view that contains the selected findings that you want to add. You can also drag findings from a findings table to the table in the Report Editor view - or in the Report Editor or directly to an existing findings category in the category tree.

Preview tab

You can preview AppScan Source for Analysis reports as you edit your templates. From the Preview pane, click **Preview** to see the report for the open assessment.

Generating custom reports

The procedures in this section's topics describe how to design and generate a report from an existing custom report. Alternatively, you can create a new report. To edit an existing report, open the report and follow the design, modification, and preview procedures.

- "Designing a report from an existing custom report"
- "Including categories in the report"
  - "Adding bundles to a category" on page 180
  - "Adding findings to a category" on page 180
  - "Adding properties to a category" on page 180
- "Previewing the report" on page 180
- "Saving the report template" on page 180

Designing a report from an existing custom report

Procedure

1. In the Report Editor view, click **New report from existing** on the toolbar.
2. Select a report template from the list of existing reports. In the Layout pane, preview the report template.
3. Change the report name, headers and footers, or template elements:
   a. Add a **Page Header** or **Page Footer**. Page headers and footers appear on each page.
   b. Add additional elements to the report. Select the report elements that you want from the **Palette** and click **Insert** (each element must be inserted separately).
   c. Delete elements from the report. Select the elements to remove from the template and click the **Remove the selected report element** toolbar button.
4. Reorder the report elements. Select an element in the preview, and click **Move the selected report element up** or **Move the selected report element down** on the toolbar to move the report element up or down.
5. Double-click an element in the Layout pane to edit it, or select the element and click **Edit the selected report element** on the toolbar.

   In the resulting dialog box, make the changes that you want. For example, to edit a text block, make changes in the Edit Text Block dialog box by modifying the label and descriptive text.

   Note: Some elements cannot be modified.

Including categories in the report

Once you define the layout, determine the categories to include in the report.

Procedure

1. In the Categories pane, click **Create a new property category**, **Create a new bundle category**, or **Create a new findings category**.
2. Name the category by typing a label for the category, a short summary of the category that can contain a count, and descriptive text.

   Promote or demote a category or subcategory using the arrow toolbar buttons.
3. Add the bundles, findings, or properties to the category.

Adding bundles to a category

Procedure

1. Open an assessment that contains bundles. If the assessment does not already contain a bundle, you cannot add a bundle to the report.
2. In the Bundles pane, click **Add a bundle** and specify the bundle or bundles to include in the category.

Adding findings to a category

Procedure

1. Open the findings view that contains the findings that you want to add. Select the findings that you want and drag them into the table of findings or to the node of the category tree in the Report Editor.
2. Alternatively, click **Add findings** on the toolbar above the findings table to include findings selected in other views. If you selected findings in multiple views, you must select the view that contains the findings added to the category.
3. Select the findings from any view that contains a findings table.

Adding properties to a category

Procedure

1. Click **Add a property** (properties include Vulnerabilities, Mechanisms, and Technologies). When you select a property, the Knowledgebase description of that property appears, when available.
2. Choose a minimum of one property and any required properties. A finding must have all properties in the **Required Properties** list to be included in the category.

   To create a subcategory, select a category and click the left or right arrow button on the toolbar.

Previewing the report

When you design a custom report, you can preview it before generating the final report. In the Preview pane, click **Preview** to display the report for the currently opened assessment.

Saving the report template

From the Report Editor view toolbar, you can click **Save** to preserve the current report template or **Save As** to save the current report template to a new file.

If you save the report template to the same directory as an application file (`.paf` or `.gaf`), it becomes available in the list of options in the Custom Report wizard and in the Report Editor view for use in subsequent scans of that application. If you save it to `<data_dir>\reports` (where `<data_dir>` is the location of your AppScan Source program data, as described in "Installation and user data file locations" on page 258), it is available for scans of any application.

Chapter 10. Customizing the vulnerability database and scan rules

This section describes how to customize the database and integrate customized vulnerabilities and other routines into scans.

There are multiple stages in the scanning process:
- A language-specific scan is run using the vulnerability database (or AppScan Source Security Knowledgebase).
- Trace is run using the vulnerability database.
- A pattern-based scan is run using scan rules from the global scan rule library.

You can use custom rules to tailor the AppScan Source Security Knowledgebase to your specific security standards and apply those standards consistently across your enterprise. You can also customize scan rules.

Extending the AppScan Source Security Knowledgebase

This section describes how to customize the database and integrate customized vulnerabilities and other routines into scans. Custom rules tailor the AppScan Source Security Knowledgebase (or vulnerability database) to your specific security standards and apply those standards consistently across your enterprise.

Often it becomes important to specify your own validation and encoding routines - or to define certain application programming interfaces (API) as vulnerabilities, sinks and sources, taint propagators, or informational items. When you create these rules, you customize and extend the AppScan Source vulnerability database, an integral part of theAppScan Source Security Knowledgebase. Once you add a custom rule to the database, AppScan Source for Analysis identifies it during a scan. Calls to the custom API are revealed as security findings or scan coverage findings - and then the findings are reported.

For example, an analyst might add an API named `readBuffer( )`, which is a `BufferOverflow` type. Subsequent scans then refer to this new API when AppScan Source for Analysis finds a vulnerability that meets its specification. For more details about vulnerability types, see the AppScan Source Security Knowledgebase (select **Help** > **Security Knowledgebase** in the main workbench menu).

When you add custom validation and encoding routines, AppScan Source for Analysis no longer treats data passed into and out of those routines as vulnerable. By adding a custom routine to the Knowledgebase, AppScan Source for Analysis determines whether data flows from a source of a tainted input to an output without validation or encoding.

Note: The AppScan Source Security Knowledgebase does not provide online help for custom records - but displays help for the vulnerability type.

Important: You must have `Knowledgebase Management` permissions to make changes to the AppScan Source Security Knowledgebase.

Creating custom rules

In the Custom Rules view, you can open the Custom Rules Wizard, a tool that guides you through the creation of custom database records. Once you create custom rules, you view them in the Custom Rules view. The table displays the signature, language, and the purpose.

Project-specific validation and encoding routines only appear in the Custom Rules view if the project to which the rules apply exists in an application under **All Applications** in the Explorer view.

- **Signature**: The signature is the fully-qualified function name. For example, the Java signature includes arguments and return types, such as `com.test.vulnerable.VulnClass.vulnerable(java.lang.string;int):int.`
- **Language**: C/C++, Java, Visual Basic, Classic ASP, or .NET
- **Purpose**: The custom record type or types on the given method, such as a `Validation.EncodingRequired` routine, sink, or source.

Tip: If you are refining your assessment of a code base by scanning iteratively and adding custom rules, and then re-scanning without changing the source code, you can dramatically reduce scan time by setting the project properties to use a vulnerability analysis cache. To do this, select the **Enable Vulnerability Analysis cache** check box in the project properties. To learn how to set project properties, see the instructions for using the "Selected project Overview tab" on page 195.

Using the Custom Rules wizard

The Custom Rules Wizard helps you add methods to the AppScan Source Security Knowledgebase. The scope of most custom rules is global (applies to all projects). Custom vulnerabilities, sources, sinks, and taint propagators are always global. Custom validation/encoding routines are not global.

Note: The Custom Rules Wizard does not validate your selections. For example, you may define a custom rule that identifies a method as a taint propagator and a sink, which is not a valid scenario.

The Custom Rules Wizard guides you through the process of defining and adding the following items to the Knowledgebase:

- Vulnerabilities
- Not a validation/encoding routine
- Sinks and sources
- Tainted callbacks
- Taint propagators
- Informational findings

Vulnerability

A vulnerability is a method that appears as a finding. A vulnerability may be a source, sink, or both.

- **Sink (Susceptible to Taint)**: An API that sends data out of the program (or out of the visible portion of the program) to a file, the network, a database, other library, or device that might be susceptible to malicious input.
- **Source (of Taint)**: Provides input to a program, which might be malformed or malicious.

Not a Validation/Encoding Routine

Marking an API as **Not a Validation/Encoding Routine** identifies that this API does not validate any data.

Tainted Callback

A callback is a routine in your code that is typically invoked by *other* code (for example from within a lower-level framework). The callback is passed as an argument to the other code, and it can be invoked at a later time with possibly tainted arguments. If you suspect that a callback could have tainted data passed to its arguments, you can mark it as a tainted callback. This causes the flow of tainted data through the routine to be visible.

A routine marked as a tainted callback will be analyzed as though it is at the root of the call graph (in other words, called by some unknown external caller) with all of its input arguments considered to be tainted. As a result, AppScan Source will report findings with traces which begin at the arguments to the tainted callback.

If the same routine is called in other contexts by your application code, it will be treated without any special considerations for taint. In those contexts, the usual analysis will take place.

Taint Propagator

Marking a method as a taint propagator implies that if any arguments to the API are derived from unvalidated input data (tainted data), then after the call, non-constant data referenced by the other arguments, as well as the return value, are also potentially tainted. Such data must be validated or encoded before being sent to a sink. This situation usually occurs because data from the tainted argument is copied or appended to the other arguments or returned.

Not Susceptible to Taint

Marking an API as not susceptible to taint (not a taint propagator), implies that calling the API with an argument derived from unvalidated input data (tainted data) does not cause the API to behave unsafely or maliciously.

If tainted data reaches a call, and the call is marked not susceptible to taint, AppScan Source ignores the call as far as tracing is concerned. AppScan Source trace does not report a lost trace and does not treat the propagated data as tainted.

Note: If tainted data reaches a method that is not a validation or encoding routine, a sink, a taint propagator, or not susceptible to taint, the method is reported as a lost trace. Non-constant arguments and return values may or may not be tainted. The AppScan Source trace call graph displays a lost trace.

Informational

Lines of code identified as informational findings may not be vulnerable but should be included in a security audit.

Adding a rule

This task topic describes the procedure for adding a custom rule using the Custom Rules Wizard.

About this task

Note: Adding or removing security or scan coverage findings and changing severity affects the project's V-Density.

Procedure

1. Open the wizard from the Custom Rules view by clicking the **Launch Custom Rules Wizard** button.
2. In the **Select Application, Project, and Files** page, select the **Application** and **Project** that the rule will apply to. Be certain that the current application and project relate to the source code of the item you want to add to the Knowledgebase. Select the **Configuration** if one is available.
3. In the **Scope** section, set the scope of the scan. Depending on the language that you are scanning, these are the scope options:

Table 21. Project file options by language

Language	Project file options
.NET	• **Scan the whole project for method signatures** • **Select one or more files external to the project** A .NET project includes any valid assembly, typically a `.dll` or `.exe` file.
Java	• **Scan the whole project for method signatures** • **Select one or more files in the project.** • **Select one or more files external to the project** A Java project includes `.jar` or `.class` files or a directory hierarchy of class files.
C/C++	• **Scan the whole project for method signatures** • **Select one or more files in the project**
Visual Basic	Scan `FRM` (forms) files, `CLS` (class) files, and `BAS` (basic)
Classic ASP	Scan ASP files only

   - **Scan the whole project for method signatures** is the default scan mode. This mode scans the entire project and returns all available signatures. This scan mode may be time consuming.
   - The **Select one or more files in the project** option isolates certain project files containing methods that might require custom rules.
   - The **Select one or more files external to the project** option identifies files external to this project to include in the scan.
4. In the **Caching** section, select the check box to reread a modified project or modified code. The vulnerability analysis cache will also be cleared (if the current project is set to cache vulnerability analysis, the vulnerability analysis cache will be re-created in the next scan).
5. **String Analysis**: String analysis monitors string manipulation in Java or Microsoft .NET projects. It provides the automatic detection of sanitizer and

validator routines. With this detection, false positives and negatives can be reduced. Select the **Enable String Analysis to find validator/sanitizer functions** check box to enable string analysis. The **Apply imported rules to Global Scope** check box determines if the discovered sanitizer or validator routines should be applied to a single project or on a global level (to all projects).

Note: The application of string analysis can slow a scan. It is therefore recommended that it should only be applied after code changes and then disabled for subsequent scans. In addition, the discovered routines should be viewed as *suggestions* and reviewed by auditors. These routines can be viewed in the Custom Rules view.

6. Click **Next** to proceed to the next page in the wizard.
7. In the **Select Methods** page:
   a. Select the method or methods to add to the Knowledgebase. The method is the name of the vulnerable API.

      The list of methods can be filtered in two ways:

      - Automatic filtering: Type the filter text in the **Filter** field. As you type, the filter is automatically applied to the list of methods. This is the default filter mode.
      - Manual filtering: Type the filter text in the **Filter** field and then click the **Filter** button (or press Enter) to apply the filter to the list. You may want to use manual filtering when large numbers of methods cause automatic filtering delays.

      In both cases, the asterisk (*) and question mark (?) characters can be used as wildcards. An asterisk matches any group of zero or more characters, while a question mark matches any single character.

      To change the filter mode, use the **Filter** button as a toggle by double-clicking it - or by using the keyboard to navigate to it and then pressing the space bar. When manual filtering is on, the **Filter** button appears not pressed and its hover help reads **Apply filter (double-click or press space to filter automatically)**. When automatic filtering is on, the button appears pressed and its hover help reads **Filter manually**.

      To better view the list of methods, expand and collapse actions are available. To expand or collapse the entire tree, right-click and select **Expand All** or **Collapse All**. To expand a package or class and all of its subentries, right-click the package or class and select **Expand Children**.

      To select multiple methods, use the keyboard Ctrl or Shift keys.

      Select the **Show full signatures** check box to display the fully-qualified signature of the methods in the tree. For example, the fully-qualified Java signature includes the package, class, method, argument types, and return types, such as `com.test.vulnerable.VulnClass.vulnerable(java.lang.string;int):int`.
   b. Identify if the scan should mark the method as:
      - "Not a Validation/Encoding Routine" on page 183
      - "Vulnerability" on page 182
      - "Tainted Callback" on page 183
      - "Taint Propagator" on page 183
      - "Not Susceptible to Taint" on page 183
      - "Informational" on page 183

8. If you are adding the method as a “Not a Validation/Encoding Routine” on page 183, “Tainted Callback” on page 183, “Taint Propagator” on page 183, or “Not Susceptible to Taint” on page 183, click **Finish** to add the records to the AppScan Source Security Knowledgebase.
9. If you are adding the method as a “Vulnerability” on page 182:
   a. Click **Next** to proceed to the next page in the wizard.
   b. In the **Assign Rule Attributes** page, select one or more method signatures and then the severity and classification. Select the check box of the vulnerability types to apply to the method or methods (multiple vulnerability types can be selected).
      - **Severity**: Level of the vulnerability's impact: **High**, **Medium**, or **Low**.
      - **Classification**: **Definitive**, **Suspect**, or **Configuration**.
      - **Vulnerability Type**: Vulnerability category.
   c. Click **Finish** to add the records to the AppScan Source Security Knowledgebase.
10. If you are adding the method as an “Informational” on page 183:
   a. Click **Next** to proceed to the next page in the wizard.
   b. In the **Assign Rule Attributes** page, select one or more method signatures. Select the check box of the vulnerability types (categories) to apply to the method or methods (multiple vulnerability types can be selected).
   c. Click **Finish** to add the records to the AppScan Source Security Knowledgebase.

Customizing input/output tracing through AppScan Source trace

Some applications (particularly web applications) require input/output tracing to identify security vulnerabilities related to SQL injection, command injection, and cross-site scripting. Through AppScan Source trace, you can specify a validation routine that, if used, eliminates the reporting of any vulnerability. All other outputs are marked as vulnerabilities if input has not been validated.

User-defined validation routines are routines that process input data and make it safe to pass to output routines. If a validation routine processes input data before passing it to an output routine, no input validation vulnerability exists. Developers may specify their own input validation and encoding routines to work with tracing.

Customizing with pattern-based scan rules

AppScan Source pattern-based scanning is an analysis of your source code based on customized search criteria. Pattern-based scanning is similar to grep (grep searches one or more files for a given character string or pattern). Auditors or security analysts performing triage might use pattern-based scanning to search for specific patterns in specific applications or in a project. Once you define a pattern as a vulnerability type, a scan of your source code identifies the pattern as a vulnerability. When AppScan Source finds a match, the item appears in the findings table. The out-of-the-box AppScan Source scan rule library includes predefined scan rules and scan rule sets (collections of scan rules).

Pattern-based scanning searches for a *regular expression*. A regular expression, often called a pattern, is a string that describes or matches a set of strings, according to certain syntax rules. You specify a search by creating a scan rule. A scan rule is similar to a custom rule that you add to the AppScan Source Security

Knowledgebase in the Custom Rules view. When you create a scan rule, you define severity, classification, vulnerability type, and other criteria.

The “Scan Rule Library view” on page 191 allows you to create new scan rules and rule sets - and you can modify or remove existing scan rules and rule sets. From the Properties view for a selected project, you can define which scan rules or scan rule sets will be used for a scan. From the Properties view for a selected application, you can define which scan rule sets will be used for a scan. You can also create new scan rules from either Properties view.

Examples of scan rules that can be created include:
- File name pattern matches
- Single rule with multiple patterns
- Absence rules

Scan rules and scan rule sets in the out-of-the-box AppScan Source scan rule library cannot be modified or removed. You must have **Manage Scan Rules** permission to be able to create scan rules or scan rule sets - or to modify and remove custom scan rules and scan rule sets.

Scan rule sets

A scan rule set is a collection of scan rules. You can add new scan rule sets or you can modify or remove existing ones. AppScan Source provides a series of language-specific scan rule sets that you can choose to apply to your projects or applications (for example, you may want to apply the **Java** scan rule set to your **Java/JSP** projects).

The “Scan Rule Library view” on page 191 allows you to create new scan rules and rule sets - and you can modify or remove existing scan rules and rule sets. From the Properties view for a selected project, you can define which scan rules or scan rule sets will be used for a scan. From the Properties view for a selected application, you can define which scan rule sets will be used for a scan. You can also create new scan rules from either Properties view.

Some of the scan rule sets that are shipped with AppScan Source contain no scan rules. You can add scan rules that are appropriate for your organization to these scan rule sets. These scan rule sets include:
- ColdFusion
- JQuery
- Client Side JavaScript
- Visual Basic 6
- MooTools

Tip: In the Scan Rule Library view, right-click a scan rule set and select **Rule Set Properties** to open a dialog box that displays information about the scan rule set. The Rule Set Properties dialog box provides information such as the number of rules in the scan rule set and parent/child relationships with other scan rule sets.

Creating a scan rule set

You must have **Manage Scan Rules** permission to be able to create scan rule sets.

Procedure

1. In the Scan Rule Library view, click **Add Scan Rule Set**.

2. In the Add Scan Rule Set dialog box, enter a name for the scan rule set in the **Name** field and then click **OK**.
3. The new scan rule set appears in the list of scan rules. You can populate the rule set in one of two ways:
   a. Select a rule or multiple rules in the scan rule section and drag and drop them into the rule set.
   b. Select a rule or multiple rules in the scan rule section and then right-click the selection and select the **Add Scan Rule to Set** menu item. In the Choose Rule Set dialog box, select the rule set that you want to add the rule or rules to.

Modifying and removing scan rule sets

Scan rule sets that you have created can be modified and removed in the Scan Rule Library view (scan rule sets in the out-of-the-box AppScan Source scan rule library cannot be modified or removed).

Important: When you modify or remove scan rule sets, you are doing so globally and for all future scans. You must have **Manage Scan Rules** permission to be able to modify or remove scan rule sets.

Modifying a scan rule set

These modifications can be made to a scan rule set:

- You can add a scan rule to an existing scan rule set by following the instructions for populating a new rule set in “Creating a scan rule set” on page 187.
- You can remove a scan rule or multiple scan rules from a scan rule set by selecting the scan rule or rules to delete and performing one of these actions:
  - Click **Remove Scan Rules From Set**.
  - Right-click and select **Remove Scan Rules From Set**.
- You can add a rule set to another rule set in one of two ways:
  - Select a rule set and drag and drop it into another rule set.
  - Right-click a rule set and select **Add Rule Set to Rule Set** and then, in the Choose Rule Set dialog box, select the rule set that you want to add as a parent rule set.

Removing a scan rule set

To remove a scan rule set, select it and perform one of these actions:

- Click **Remove Scan Rule Set**.
- Right-click and select **Remove Scan Rule Set**.

Searching for text patterns

Within a given source file, pattern-based scanning searches for text patterns in files by extension, allowing the search to exist in source files, XML configuration files, and other text files.

For example, you might want to create a pattern search to ensure that improper email addresses are not hardcoded into your application. In this case, if you want to make sure that the application does not use corporate email addresses, you could search for a pattern such as `.*@mycompany.com`.

Examples

This pattern finds	Pattern
An email address	[A-Za-z]\.[A-Za-z]@[A-Za-z][A-Za-]\.com
All instances of the pattern, such as passWord =	[Pp][Aa][Ss][Ss][Ww][Oo][Rr][Dd]\W*=
Any instance of the MD5 hashing algorithm	getInstance[[:space:]]*\ ([[:space:]]*"MD5

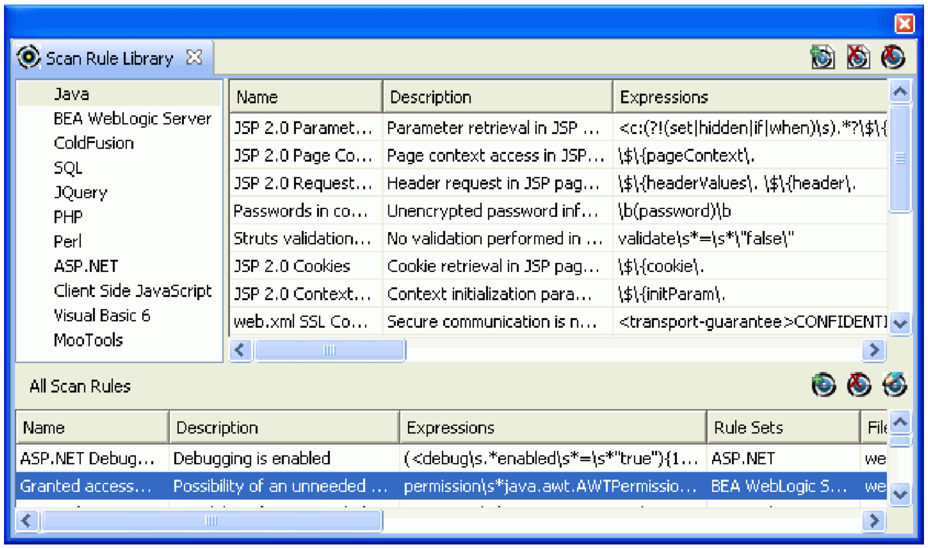
Defining scan rules for analysis

AppScan Source text scan rules can be Extended Global Regular Expressions Print (egrep), Global Regular Expressions (grep), or Perl regular expressions. These regular expressions (expressions that have string values that use the complete set of alphanumeric and special characters) match the scan rules.

Character	Description
^	Starts with
$	Ends with
\n, \t, or \r	Literal newline, tab, return
[xyz]	Any characters listed
[^abx]	Any characters except those listed
[a-fA-F0-9]	Any hex character
.	Any character
|	Either
\	Cancel special character meaning \$ \^ \\ \?

Scan rules are stored in a global scan rule library (on the AppScan Source server) and may be shared across projects and applications. Scan rules and scan rule sets can also be shared by all users. Scan rules are added by reference, which you can disable by removing the reference in the associated object without deleting the underlying scan rule.

You create scan rules in the Scan Rule Library view or the Properties tab of the Explorer view. When you install AppScan Source, the Scan Rule Library view displays AppScan Source-provided rules. In this view, you can edit, delete, or create a rule.



Scan Rule Library

Java
BEA WebLogic Server
ColdFusion
SQL
JQuery
PHP
Perl
ASP.NET
Client Side JavaScript
Visual Basic 6
MooTools

Name	Description	Expressions
JSP 2.0 Paramet...	Parameter retrieval in JSP ...	<c:(?!(set|hidden|if|when)\s).*?\$\{
JSP 2.0 Page Co...	Page context access in JSP...	\$\{pageContext\.
JSP 2.0 Request...	Header request in JSP pag...	\$\{headerValues\. \$\{header\.
Passwords in co...	Unencrypted password inf...	\b(password)\b
Struts validation...	No validation performed in ...	validate\s*=\s*\"false\"
JSP 2.0 Cookies	Cookie retrieval in JSP pag...	\$\{cookie\.
JSP 2.0 Context...	Context initialization para...	\$\{initParam\.
web.xml SSL Co...	Secure communication is n...	<transport-guarantee>CONFIDENTI

All Scan Rules

Name	Description	Expressions	Rule Sets	File
ASP.NET Debug...	Debugging is enabled	(<debug\s.*enabled\s*=\s*"true"){1...	ASP.NET	we
Granted access...	Possibility of an unneeded ...	permission\s*java.awt.AWTPermissio...	BEA WebLogic S...	we

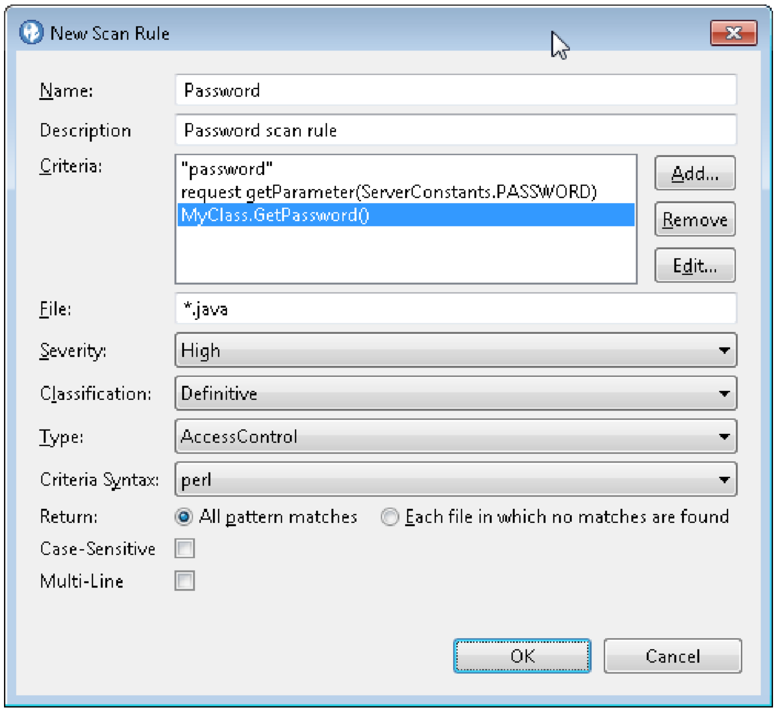
Important: You can add or remove search criteria, but each pattern-based rule must have at least one search criteria.

Creating a scan rule in the Scan Rule Library view

You must have **Manage Scan Rules** permission to be able to create scan rules.

Procedure

1. In the Scan Rule Library view, click **Add Scan Rule**.
2. In the New Scan Rule dialog box, **Name** the scan rule.
3. Optional: Add a **Description** for the scan rule.
4. Add the **Criteria**. Click **Add** and type the regular expression for each rule.
5. Identify the file type, such as such as `*.java` or `*.xml`. You can type any file type with or without wildcard characters.
6. Select the **Severity**:
   - **High**
   - **Medium**
   - **Low**
   - **Info**
7. Select the **Classification**:
   - **Definitive**
   - **Suspect**
   - **Scan Coverage**
8. Select the vulnerability type to search for in the scan. (For more details about vulnerability types, see the AppScan Source Security Knowledgebase)



Name: Password

Description: Password scan rule

Criteria:
"password"
request.getParameter(ServerConstants.PASSWORD)
MyClass.GetPassword()

File: *.java

Severity: High

Classification: Definitive

Type: AccessControl

Criteria Syntax: perl

Return: [✓ All pattern matches] Each file in which no matches are found

Case-Sensitive ☐

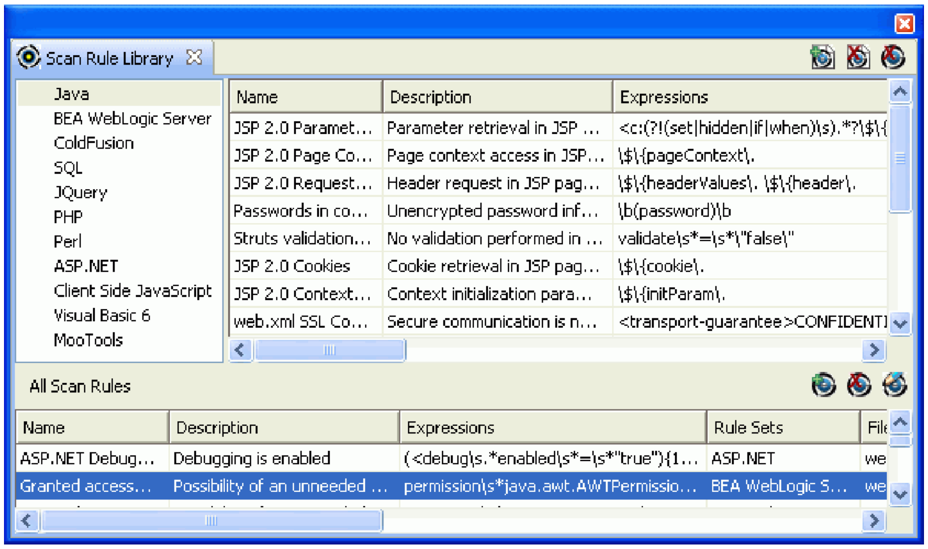
Multi-Line ☐

9. Select the criteria syntax:
   - **egrep**
   - **grep**
   - **perl**
10. Identify if the results returned include **All pattern matches** or **Each file in which no matches are found.** (When no matches are found, the pattern is an absence rule.)
11. Click **OK** to verify that the regular expressions in the scan rule are valid. The scan rule is then added to the scan rule library.

Scan Rule Library view:

Pattern-based scanning is an analysis of your source code based on customized search criteria. The Scan Rule Library view allows you to view existing pattern-based scan rules, by language (including the out-of-the-box AppScan Source scan rule library). In addition, the view allows you to add rules and patterns for pattern-based scanning.

Once you build a scan rule library, you can apply the pattern analysis to specific applications or projects. See "Customizing with pattern-based scan rules" on page 186 for details about pattern searches.



Java
BEA WebLogic Server
ColdFusion
SQL
JQuery
PHP
Perl
ASP.NET
Client Side JavaScript
Visual Basic 6
MooTools

Name	Description	Expressions
JSP 2.0 Paramet...	Parameter retrieval in JSP ...	<c:(?!(set|hidden|if|when)\s).*?\$\{
JSP 2.0 Page Co...	Page context access in JSP...	\$\{pageContext\.
JSP 2.0 Request...	Header request in JSP pag...	\$\{headerValues\. \$\{header\.
Passwords in co...	Unencrypted password inf...	\b(password)\b
Struts validation...	No validation performed in ...	validate\s*=\s*\"false\"
JSP 2.0 Cookies	Cookie retrieval in JSP pag...	\$\{cookie\.
JSP 2.0 Context...	Context initialization para...	\$\{initParam\.
web.xml SSL Co...	Secure communication is n...	<transport-guarantee>CONFIDENTI

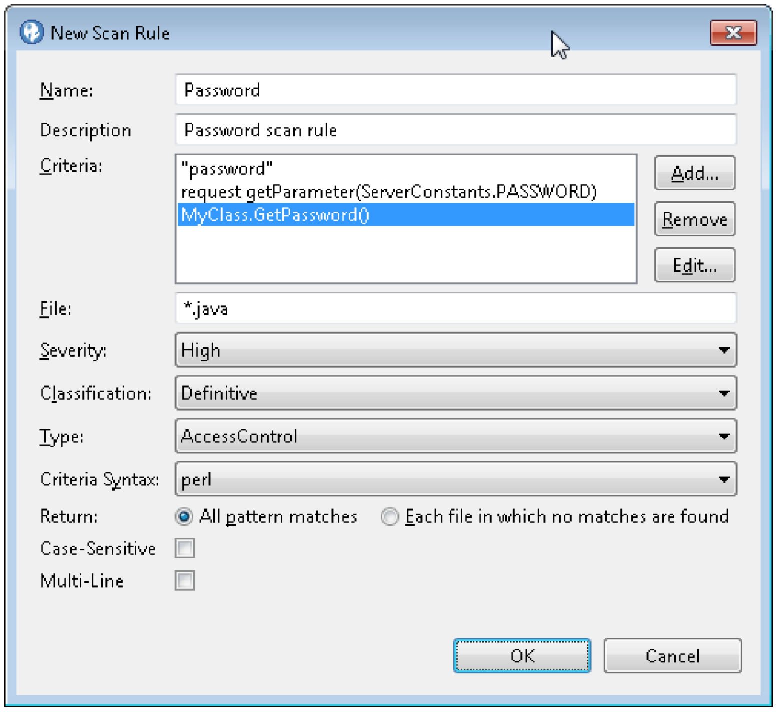
All Scan Rules

Name	Description	Expressions	Rule Sets	File
ASP.NET Debug...	Debugging is enabled	(<debug\s.*enabled\s*=\s*"true"){1...	ASP.NET	we
Granted access...	Possibility of an unneeded ...	permission\s*java.awt.AWTPermissio...	BEA WebLogic S...	we

Creating a scan rule in the Explorer view

Procedure

1. In the Explorer view, select the project to which the scan rule analysis applies. On the **Scan Rules and Rule Sets** tab of the Properties view, click **Add a new scan rule**.
2. Click **New Scan Rule** in the Choose Scan Rule dialog box, and name the pattern.
3. In the New Scan Rule dialog box, **Name** the scan rule.
4. Optional: Add a **Description** for the scan rule.
5. Add the **Criteria**. Click **Add** and type the regular expression for each rule.
6. Identify the file type, such as such as `*.java` or `*.xml`. You can type any file type with or without wildcard characters.
7. Select the **Severity**:
   - **High**
   - **Medium**
   - **Low**
   - **Info**
8. Select the **Classification**:
   - **Definitive**
   - **Suspect**
   - **Scan Coverage**
9. Select the vulnerability type to search for in the scan. (For more details about vulnerability types, see the AppScan Source Security Knowledgebase)


New Scan Rule

Name: Password

Description: Password scan rule

Criteria:
"password"
request.getParameter(ServerConstants.PASSWORD)
MyClass.GetPassword()

File: *.java

Severity: High

Classification: Definitive

Type: AccessControl

Criteria Syntax: perl

Return: [✓ All pattern matches] Each file in which no matches are found

Case-Sensitive: ☐

Multi-Line: ☐

10. Select the criteria syntax:
    - **egrep**
    - **grep**
    - **perl**
11. Identify if the results returned include **All pattern matches** or **Each file in which no matches are found.** (When no matches are found, the pattern is an absence rule.)
12. Click **OK** to verify that the regular expressions in the scan rule are valid. The scan rule is then added to the scan rule library.

Results

After you create a scan rule and scan the project, the results appear as findings. From the findings, you can find the location in the code and make the appropriate modifications.

Properties view: selected application:

In this view, you configure attributes for the selected application. Application attributes depend on previously-created global attributes.
- "Overview"
- "Exclusions and Filters" on page 194
- "Scan Rules and Rule Sets" on page 194
- "Modified Findings" on page 194
- "Custom Findings" on page 195

Overview

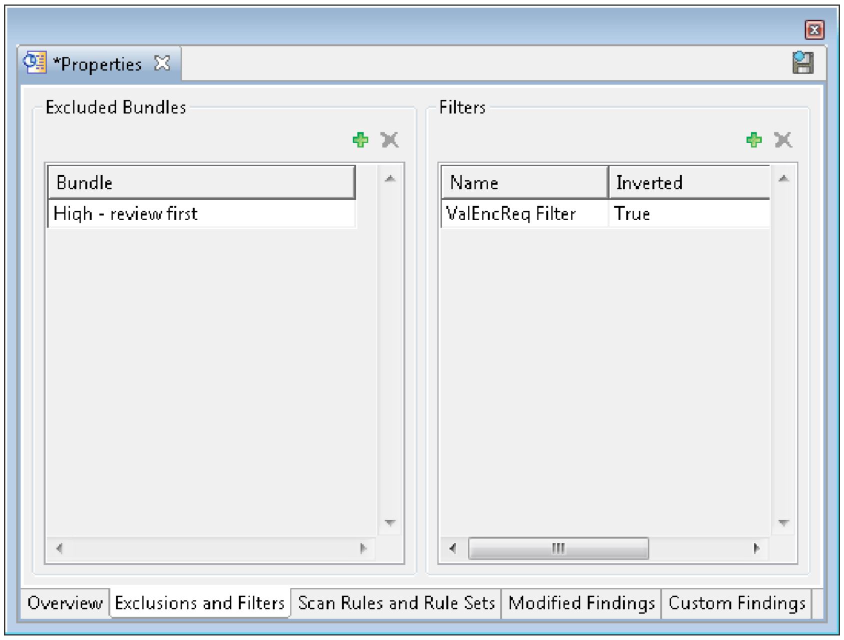
The Overview tab displays:

- The application name. The application can be renamed by entering a new name in the field.
- Application attributes

Exclusions and Filters

This tab allows you to specify existing filters for the selected application, and how you want the filters applied (a filter can be applied directly - or its inverse can be applied). In the tab, you can also manage bundles that exclude results from your scan. See Chapter 5, “Triage and analysis,” on page 99 for information about filters - and “Applying filters globally” on page 118 for details about applying them globally.

Excluded and filtered findings do not appear in scan results or factor into application or project metrics.



Bundle
High - review first

Name	Inverted
ValEncReq Filter	True

Scan Rules and Rule Sets

When you select an application in the Explorer view, the Scan Rules and Rule Sets tab in the Properties view allows you to manage application scan rules. Using pattern-based scanning, you search for text patterns that you want to appear as findings. When you create a scan rule set, you can apply it to an application or to a project (individual scan rules can only be applied to projects). See “Customizing with pattern-based scan rules” on page 186 for details about pattern-based analysis.

Modified Findings

On the Modified Findings tab, you view, edit, or delete any previously modified findings, or modify an existing finding. *Modified findings* are findings with altered vulnerability type, severity, classification, or notes.

Custom Findings

On the Custom Findings tab, you view, add, edit, or delete custom findings. See "Custom findings" on page 132 for more details.

Properties view: selected project:

In this mode of the Properties view, you configure parameters for the selected project. Project attributes depend on previously created global attributes. Properties vary according to the selected project.
- "Selected project Overview tab"
- "Filters" on page 196
- "Scan Rules and Rule Sets" on page 196
- "File Extensions" on page 196
- "Sources" on page 197
- "JavaServer Page (JSP) Project Dependencies" on page 198
- "Project Dependencies" on page 198
- "Compilation" on page 198
- "Optimizations" on page 199
- "Precompile Tab (ASP.NET only)" on page 199

Selected project Overview tab

The Overview tab displays:
- Project **Name**. The project can be renamed by entering a new name in the field.
- Project **File** name and path
- **Project Type**
- Configuration: This section displays the target configuration. For .NET and C++ projects, this section displays the target configuration that has been saved in the Project Dependencies tab. For all other project types, this section displays **Default**.
- Filter option: Select **Filter findings contained in external sources** to filter out any findings discovered in files that are not source files of the scanned project. This option reduces noise for projects where findings are reported in compiler-generated or temporary files, such as `ASP.NET`.
- Vulnerability analysis cache options: If you are refining your assessment of a code base by scanning iteratively and adding custom rules, and then re-scanning without changing the source code, you can dramatically reduce scan time by setting the project properties to use a vulnerability analysis cache. To do this, select the **Enable Vulnerability Analysis cache** check box in the project properties. The first time you scan the project after selecting this check box, a vulnerability analysis cache will be created. For every subsequent scan of the project, the vulnerability analysis cache will be used and scan time will be reduced.

  To clear the vulnerability analysis cache, click **Clear cache**. The next time you scan the project, a new vulnerability analysis cache will be created (provided that you do not deselect the **Enable Vulnerability Analysis cache** setting). You may want to clear the cache if:
  - Source code in the project has changed since the last scan.
  - You have made project configuration changes, such as adding or deleting source files.

  – You have changed code configuration options. For example, you may want to clear the cache if you are scanning Java and the class path has changed - or if you are scanning C or C++ and you have changed the `include` path or preprocessor definitions.

  Note: You can also clear the vulnerability analysis cache by selecting the **Clear cache** check box when creating custom rules in the Custom Rules Wizard.

- **String Analysis**: String analysis monitors string manipulation in Java or Microsoft .NET projects. It provides the automatic detection of sanitizer and validator routines. With this detection, false positives and negatives can be reduced. Select the **Enable String Analysis to find validator/sanitizer functions** check box to enable string analysis. The **Apply imported rules to Global Scope** check box determines if the discovered sanitizer or validator routines should be applied to a single project or on a global level (to all projects).

  Note: The application of string analysis can slow a scan. It is therefore recommended that it should only be applied after code changes and then disabled for subsequent scans. In addition, the discovered routines should be viewed as *suggestions* and reviewed by auditors. These routines can be viewed in the Custom Rules view.
- **File Encoding**: The character encoding of files in your project must be set so that AppScan Source can read the files properly (and, for example, display them correctly in the source view).

  Note: The default file encoding for AppScan Source projects is **ISO-8859-1**. The default file encoding can be changed in the General preference page.

Filters

This tab allows you to specify existing filters for the selected project, and how you want the filters applied (a filter can be applied directly - or its inverse can be applied). See Chapter 5, “Triage and analysis,” on page 99 for information about filters - and “Applying filters globally” on page 118 for details about applying them globally.

Scan Rules and Rule Sets

When you select a project in the Explorer view, the Scan Rules and Rule Sets tab in the Properties view allows you to manage application scan rules. Using pattern-based scanning, you search for text patterns that you want to appear as findings. When you create a scan rule, you can apply it to an application or to a project. See “Customizing with pattern-based scan rules” on page 186 for details about pattern-based analysis.

File Extensions

Use this tab to configure or add valid file extensions for the project - and to exclude files from scans and specify extensions as web files.

The **File Extensions** section lists the extensions that have been set globally in the “Project file extensions” on page 76 preference page for the current project type (you can choose file extensions for a different project type using the **File Extension Set** menu). To exclude an extension from scans of the current project, select it in the list and click **Exclude Extension**. This causes the extension to be listed in the **Excluded Extensions** section of the tab.

To add an additional extension for the project, select **Add Extension** in the **Additional Extensions** section, and then enter the file extension and indicate if files with the extension should be scanned, considered as a web file, or excluded.

Table 22. File extension settings

Setting	Description	Usage examples
Scan or **Assess**	Include files with the indicated extension in full analysis.	• If a `.xxx` extension is created for Java projects and marked as **Scan** or **Assess**, files with that extension will be compiled and scanned. • A file can be part of a project but not marked as **Scan** or **Assess** if it should not be compiled and scanned (such as header files in C++). These files would be included in the project and searched during pattern-based analysis.
Web File	Mark files with the indicated extension for JSP compilation. This setting allows AppScan Source to separate web sources from non-web sources.	If a `.yyy` extension is created for Java projects and marked as a **Web File**, files with that extension will be arranged as web sources in the projects. When AppScan Source prepares for analysis, these files will be precompiled into classes to be analyzed.
Exclude	Do not create source files in the project for files with the indicated extension. Files with this extension will not be scanned.	Create a `.zzz` extension for files that are necessary to your projects for compilation, but do not need to be included in analysis.

Sources

Specify the sources to include in the scan.

- Working Directory: The location of the AppScan Source project file (`ppf`) and the base for all relative paths.
- **Add Source Root** and **Remove Source Root**: The Sources tab displays the properties established for the project from the Project Configuration Wizard or defined in the imported `ppf`.

  Remove Source Root is available only when the **Source Root** icon is selected. It is used to remove the source code root directory.
- Find Source Roots (Java projects only): Allow AppScan Source for Analysis to find all valid source roots automatically.
- Project files are listed under the **Source Root** icon. Files that are excluded from scanning have a red file icon (if an excluded file is right-clicked, its menu has **Exclude** disabled and **Include** enabled). To exclude an included file, right-click it and choose **Exclude** in the menu. To include an excluded file, right-click it and choose **Include** in the menu.

JavaServer Page (JSP) Project Dependencies

The JSP Project Dependencies tab displays the properties established for the specified JSP project.

- Contains web (JSP) content: Identifies if the project is a web application that contains JavaServer Pages.
- Web context root: A `WAR` file or a directory that contains the `WEB-INF` directory. The web context root must be the root of a valid web application.
- JSP Compiler: Tomcat 7 (Jasper 2) is the default JSP compiler setting (the default JSP compiler can be changed in the Java and JSP preference page). To learn about the compilers that are supported by AppScan Source, see http://www.ibm.com/support/docview.wss?uid=swg27027486.

  Apache Tomcat Versions 5, 6, and 7 are included in the installation of AppScan Source. If the **Tomcat 5**, **Tomcat 6**, and **Tomcat 7** preference pages are not configured, AppScan Source will compile JSP files using the supplied Tomcat JSP compiler that is currently marked as default. If you want to employ an external Tomcat compiler, use the Tomcat preference pages to point to your local Tomcat installation.

  If you are using Oracle WebLogic Server or WebSphere Application Server, you must configure the applicable preference page to point to your local installation of the application server so that it can be used for JSP compilation during analysis. If you have not already completed this configuration, you will be prompted by a message to do so when you select the JSP compiler. If you click **Yes** in the message, you will be taken to the appropriate preference page. If you click **No**, a warning link will display next to the JSP compiler selection (following the link will open the preference page).

Project Dependencies

The Project Dependencies tab displays project properties. Configuration settings on this tab vary by language, for example:

- **Options** allow you to select any additional required compiler parameters.
- JDK settings are specific to Java.
- Preprocessor definitions are specific to C/C++ code.
- The target configuration is available only for .NET and C++ projects.

Compilation

- Options: Additional required compiler parameters for the project configuration.
- Use JDK: Identify the JDK used for the project compilation, as configured in Preferences. See Chapter 3, “Preferences,” on page 63.

  Java projects may refer to a local Java Development Kit (JDK) location. When projects move to the server, the JDK path may no longer be valid. To transfer local projects to the server, you must identify the default JDK path for each project that specifies a named JDK.

  Note: The default compiler for JSP projects is Tomcat 7 (Jasper 2), which requires Java Version 1.6 or higher. If **Tomcat 7** is kept as default, selecting an earlier JDK (for example, **IBM JDK 1.5**) will result in compilation errors during scans.
- Validate: **Validate** assures that project dependencies are correctly configured. It checks Java projects for configuration conflicts between sources and the class

path and for compilation errors. **A conflict exists** if a class in a class path is duplicated in a source root. (If a conflict exists, modify the class path to remove the conflicting class.)

After checking for conflicts, **Validate** determines if the project compiles and reports any compilation errors.

Optimizations

- **Precompiled classes**: Use precompiled class files instead of compiling during the scan. When selected, this option disables the source stage options.
- Stage sources files to minimize effects of compile errors: Controls whether AppScan Source copies the sources to the staging directory.

  Correct for packages not matching the directory requires Java Compile to open each source file.

  Clean staging area between scans increases performance between scans.

Precompile Tab (ASP.NET only)

Precompilation is accomplished by making an HTTP request to a special page (by default, `precompile.axd`) in the website. This page is processed by a special HTTP handler specified in the `web.config`. This handler compiles the entire site, including `client.aspx` files, to the Temporary ASP.NET Files directory in the .NET framework directory, where they are all then scanned.

To scan ASP.NET 1.1, you must instrument the website such that the website compiles and builds debug information. Subsequently, the fact that a website compiles and builds debug information is, in and of itself, a security vulnerability. You can safely ignore this vulnerability as the scan requires it. However, ensure that your deployed application is not compiled with `debug=true` in the `web.config`.

To precompile an ASP.NET 1.1 website, add this element as a child to the `<system.web>` element in your `web.config` file:

```
<httpHandlers><add verb="*" path="precompile.axd"
type="System.Web.Handlers.BatchHandler"/></httpHandlers>
```

You should also set `debug=true` in the compilation element. For example:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
 <system.web>
  <httpHandlers><add verb="*" path="precompile.axd"
  type="System.Web.Handlers.BatchHandler"/>
  </httpHandlers>
   <compilation
    defaultLanguage="c#"
    debug="true"
   />
...
```

This element specifies to the website that the page, `precompile.axd` will be handled by the special .Net `System.Web.Handlers.BatchHandler` class. This class precompiles the contents of the website to the Temporary ASP.NET Files directory.

- Website: Request target to precompile the site. The default location is `precompile.axd`. `Precompile.axd` is a virtual file and maps to the file specified in the `web.config` file.
- Output directory: The directory targeted by the precompilation. AppScan Source finds the precompilation output in this directory.

- Precompile the ASP.NET website: AppScan Source automatically precompiles and scans the precompiled output during a scan.
- Stop scan if precompile fails: Selecting **Precompile the ASP.NET website** and **Stop scan if precompile fails**, stops a scan if precompilation fails. Otherwise, the scan continues only on the website's primary output.
- Compile Now: Test to see that the precompilation, based on the current settings, succeeds before scanning. Compilation output displays in the Precompile Output pane.
- Additional Assemblies: For any .NET project type, specify additional assemblies to scan.
- Project References: List the directories in which to search for referenced assemblies in .NET assembly projects and existing .NET projects.

Modifying and removing scan rules

Scan rules that you have created can be modified and removed in the Scan Rule Library view (scan rules in the out-of-the-box AppScan Source scan rule library cannot be modified or removed).

Important: When you modify or remove scan rules, you are doing so globally and for all future scans. You must have **Manage Scan Rules** permission to be able to modify or remove scan rules.

Modifying a scan rule

To edit a scan rule, select it and perform one of these actions:
- Click **Edit Scan Rule**.
- Right-click and select **Edit**.

This opens the Edit Scan Rule dialog box, which allows you to modify any setting but the scan rule name.

Removing a scan rule

Select one or more scan rules and perform one of these actions:
- Click **Remove Scan Rule**.
- Right-click and select **Delete**.

Applying scan rules and scan rule sets

Scan rules and scan rule sets are applied at the project level in the Properties view. Scan rule sets can also be applied at the application level in the Properties view. After you scan applications or projects with applied scan rules, the results of the scan rule search appear in the views that contain findings.

Note: Individual scan rules assigned at the application level have no affect on scan results. In order to assign scan rules to an application, they must first be grouped in to a scan rule set, which can then be assigned to the application.

Applying scan rules and scan rule sets to projects

If scan rule sets have been set for the application that contains your project, those scan rules will be applied during scans of the project. You do not need to modify project properties to add the scan rule settings that have been made at the application level.

Applying scan rules and scan rule sets

Select the project or application in the Explorer view and then make the modifications listed below to the Scan Rules and Rule Sets tab of its Properties view. After specifying the scan rules and rule sets to apply to the application or project, save the application or project properties. Subsequent scans of the application or project will include these scan rules.

- To apply a scan rule set, click **Add a new scan rule set** or right-click in left hand frame of the Scan Rule Sets section and click **Add a new scan rule set**. In the Choose Rule Set dialog box, select the rule set that you want to apply and then click **OK**. The scan rules that are already defined as part of that scan rule set are then displayed in the right hand side of the Scan Rule Sets section.
- To remove a scan rule set so that it is not used during scans of the application or project, select it and perform one of these actions:
  - Click **Remove or exclude the selected scan rule set**.
  - Right-click and select **Remove or exclude the selected scan rule set**.
- You can apply individual scan rules to projects (in addition to scan rules that are already being applied because they're in a scan rule set that is being applied). To apply a scan rule or rules, select the project and click **Add a new scan rule** or right-click in the Scan Rules section and select **Add a new scan rule** from the menu. Select from existing scan rules in the Choose Scan Rule dialog box (you can select multiple scan rules) - or create a new scan rule following the instructions in "Creating a scan rule in the Explorer view" on page 192.
- If scan rules have already been added, you can remove them so they are no longer applied during scans of the application or project. To remove a scan rule or rules from a rule set, select one or more scan rules and perform one of these actions:
  - Click **Remove or exclude the selected scan rules**.
  - Right-click and select **Remove or exclude the selected scan rules**.

Chapter 11. AppScan Source for Analysis samples

AppScan Source for Analysis includes sample applications that you can use to familiarize yourself with the product.

After AppScan Source for Analysis is installed, the sample applications are located in `<data_dir>\samples` (where `<data_dir>` is the location of your AppScan Source program data, as described in "Installation and user data file locations" on page 258).

Sample Java application: `simpleIOT`

The `simpleIOT` sample is a small Java application that contains various security vulnerabilities. It can be manually imported to the AppScan Source for Analysis workbench - or you can import the application file (`SimpleIOT.paf`) or project file (`SimpleIOT.ppf`) that are included with the sample. To learn how to add applications and projects, see Chapter 2, "Configuring applications and projects," on page 17.

Once you have added the sample to AppScan Source, you can scan it and explore its findings.

Sample application for learning the Framework for Frameworks handling APIs: `F4FEjbExample.zip`

This example project archive is used for demonstrating the Framework for Frameworks handling APIs. See the *IBM Security AppScan Source Utilities User Guide* for more information.

Chapter 12. The AppScan Source for Analysis work environment

To get the most out of AppScan Source, you should understand the basic concepts behind the AppScan Source for Analysis working environment and how to use the options that best fit your workflow.

The AppScan Source for Analysis workbench

AppScan Source for Analysis workflow occurs in a *workbench*, which consists of perspectives, views, and editors that display or are hidden, depending on context.

Perspectives

The three perspectives in the product - Configuration, Triage, and Analysis - consist of multiple views. Although each perspective opens with default views, you can reorganize views to customize each perspective. The views are described in detail in the Chapter 13, “Views,” on page 217 section of the help.

- Configuration Perspective: Create and manage applications, projects, and attributes.
- Triage Perspective: View scan results to prioritize remediation workflow and separate real vulnerabilities from potential ones. This perspective can be used to isolate the issues that you need to fix first.
- Analysis Perspective: Drill down into individual findings - and review source code, remediation advice, and AppScan Source trace information.

Workbench window

The AppScan Source for Analysis workbench window consists of these elements:

- Main menu: Menus that access AppScan Source for Analysis functions
- Toolbar: Icons and buttons for frequently-used functions
- Perspectives: Collections of views
- Views: Presentations and ways to navigate the information in the workbench

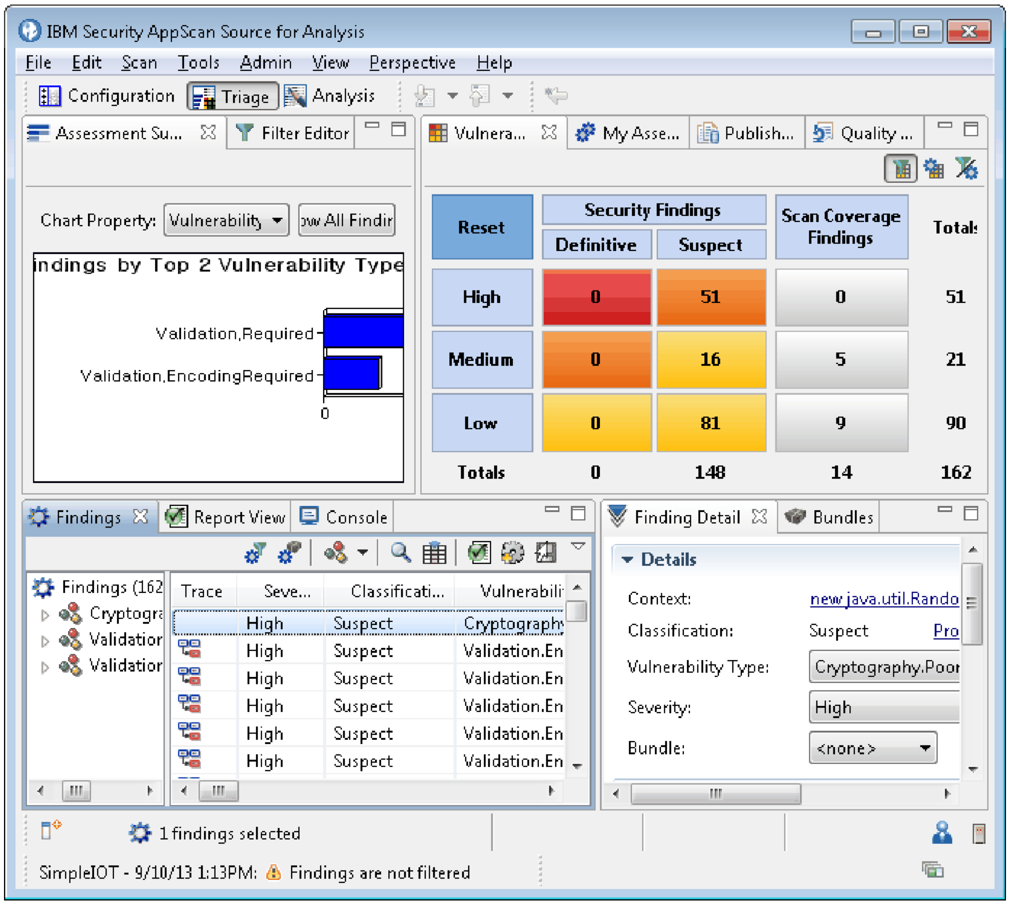


Chart Property: Vulnerability ▾ ow All Findir

indings by Top 2 Vulnerability Type

Validation.Required

Validation.EncodingRequired

0

	Security Findings		Scan Coverage Findings	Totals
	Definitive	Suspect		
High	0	51	0	51
Medium	0	16	5	21
Low	0	81	9	90
Totals	0	148	14	162

Findings (162
- Cryptogra
- Validatior
- Validatior

Trace	Seve...	Classificati...	Vulnerabili
	High	Suspect	Cryptography
	High	Suspect	Validation.En
	High	Suspect	Validation.En
	High	Suspect	Validation.En
	High	Suspect	Validation.En
	High	Suspect	Validation.En

Context:	new java.util.Rando	
Classification:	Suspect	Pro
Vulnerability Type:	Cryptography.Poor	
Severity:	High	
Bundle:	<none>	

1 findings selected

SimpleIOT - 9/10/13 1:13PM: Findings are not filtered

Toolbars and information at the bottom of the workbench

- **Fast View** toolbar: Fast views are hidden views that can be quickly opened and closed. They work like other views except they do not take up space in your workbench window. Fast views are represented by toolbar buttons on the fast view bar, which is the toolbar on the bottom left of the workbench window. When you click the toolbar button for a fast view, that view opens temporarily in the current perspective (overlaying it). As soon as you click outside that view or the view loses focus it is hidden again. To set a view as a fast view, click **Show View as a Fast View** and then choose the view from the menu.
- Selected findings: When findings are selected, an indicator at the bottom of the workbench displays the number of selected findings.
- Source file information: When a source file is open, this information about the file displays at the bottom of the workbench:
  – Whether the file is writable or read-only. If you attempt to edit a read-only file, a prompt in AppScan Source for Analysis will allow you to set the file to writable.
  – If your operating system input mode is insert or overwrite.
  – The current cursor location in the file (line and column number).
- Server connection information: Hovering over the user icon indicates the user that is currently logged in to the AppScan Enterprise Server - and hovering over the server icon allows you to see the AppScan Enterprise Server that AppScan Source for Analysis is connected to.
- When an assessment is open, the bottom of the workbench includes this information:

  - The name of the assessment, and the date and time of its creation.
  - An indicator that allows you to quickly determine how filters have been applied to the findings in the assessment. See "Determining applied filters" on page 119 for more information.
- A progress indicator is also displayed at the bottom of the workbench that indicates actions in progress. For example, this indicator appears during scans and assessment publication. In addition, this section indicates when an assessment is open.

Main menu

The main menu bar contains menus that allow you to perform a variety of actions. Your user privileges may regulate the commands that are available to you in these menus.

- "File menu"
- "Edit menu" on page 211
- "Scan menu" on page 212
- "Tools menu" on page 212
- "Admin menu" on page 213
- "View menu" on page 213
- "Perspective menu" on page 213
- "Help menu" on page 214

File menu

The **File** menu offers options for applications, projects, and assessments - and allows you to exit the product. Some **File** menu items are context-sensitive and depend on the active view and the currently-selected item in that view.

Table 23. File menu

Menu item	Description	Keyboard shortcut
Add Application > **Create a new application**	Add a new application to the set of applications. This action launches the New Application Wizard.	Ctrl+N
Add Application > **Open an existing application**	This launches an Open dialog box, which allows you to browse for and add an existing application to the set of applications. File types that can be added include `.paf`, `.sln`, `.dsw`, and `.ewf`.	Ctrl+O

Table 23. File menu (continued)

Menu item	Description	Keyboard shortcut
Add Application > Import an existing Eclipse/ RAD workspace	This launches the Add Workspace dialog box, which allows you to add an existing Eclipse or IBM Rational Application Developer for WebSphere Software (RAD) workspace that contains Java projects. After the workspace has been imported, you will be able to scan any Java projects that it contains. **Note:** Before importing a workspace, be certain that you have installed and updated the development environment as described in "Configuring your development environment for Eclipse and Rational Application Developer for WebSphere Software (RAD) projects" on page 26.	
Add Application > Multiple Applications	Add multiple applications to the set of applications. This action launches a dialog box that allows you to specify a directory in which to search for applications. In the search results, you can select one or more applications to add.	
Add Application > Discover Applications	This launches the Application Discovery Assistant, which allows to you to quickly create and configure applications and projects for Java and Microsoft Visual Studio source code.	
Remove Application	If an application is selected in the Explorer view, this action is available and choosing it removes the selected application.	
Add Project > New Project	If an application is selected in the Explorer view, this action is available and choosing it allows you to add a new project to the application. This action launches the New Project Wizard.	

Table 23. File menu (continued)

Menu item	Description	Keyboard shortcut
Add Project > **Existing Project**	If an application is selected in the Explorer view, this action is available and choosing it allows you to add an existing project to the application. This action launches a dialog box that allows you to browse for a `.ppf` , `.vcproj`, `.vcxproj`, `.csproj`, `.vbproj`, `.dsp`, or `.epf` file to open.	
Add Project > **Copy Project**	If a project is selected in the Explorer view, this action is available and choosing it opens a dialog box that allows you to copy the project to another application - or create a copy of the project in the application that currently contains the project.	
Add Project > **Multiple Projects**	Add multiple projects to the application that is selected in the Explorer view. This action launches a dialog box that allows you to complete one of these tasks: • Specify a directory in which to search for projects. • Specify a workspace in which to search for projects. • Specify a Microsoft Solution file in which to search for projects. In the search results, you can select one or more projects to add.	
Register	Register the selected application or project with AppScan Source. You must register applications and projects before they can be published to the AppScan Source Database.	
Unregister	Unregister the selected application or project.	
Open Assessment	This launches an Open dialog box, which allows you to browse for an AppScan Source assessment file. File types that can be opened include `.ozasmt` and `.xml`.	F7

Table 23. File menu (continued)

Menu item	Description	Keyboard shortcut
Close Assessment	Close the assessment that is currently open in the Triage perspective.	
Save Assessment	Save the open assessment to a file.	Ctrl+Shift+S
Save Assessment As	Save the assessment with a different name, save it in a different directory, or both.	
Publish Assessment to AppScan Source	Store the current assessment in the AppScan Source Database. The application that was scanned (or the project or file that the application contains) must be registered before the publish action can be completed. If the application has not been registered, you will be prompted to register when you choose the publish action.	
Publish Assessment to AppScan Enterprise Console	If your AppScan Enterprise Server has been installed with the Enterprise Console option, you can publish assessments to it. The AppScan Enterprise Console preference page must be completed with valid values before you can publish assessments to the Enterprise Console.	
Save	This action is available in these circumstances: • An application's properties have been modified in the Properties view. • A project's properties have been modified in the Properties view. • A file that is open in the internal editor has been modified. Select this action to save these changes.	Ctrl+S
Exit	Exit AppScan Source for Analysis.	

Edit menu

This menu offers standard modification and search/replace controls. This menu is also used for launching product preferences. Some **Edit** menu items are context-sensitive and depend on the active view and the currently-selected item in that view.

Table 24. Edit menu

Menu item	Description	Keyboard shortcut
Cut	Copy and remove selected text. Use this action for text that is selected in the console, editor, or various text fields.	Ctrl+X
Copy	Copy selected text to the clipboard. Use this action for text that is selected in the console, editor, or various text fields.	Ctrl+C
Paste	Paste text that has been copied or cut. This action is typically used for Duplicating information and reproducing it in another part of the product.	Ctrl+V
Rename	Rename the selected object. Objects that can be renamed include applications, projects, assessments, and bundles.	F2
Remove	Remove the selected object.	Delete
Select All	Select the entire body of text. Use this action for text in the console, editor, or various text fields.	Ctrl+A
Refresh	Refresh the contents of a selected application, project, or view.	F5
Find	Search for text in the console or editor - or search for findings in a findings table.	Ctrl+F
Find Next	If the find action was used to search for text in the console or editor, use this action to find the next instance of the text.	F3
Preferences	Select this to open the Preferences dialog box. Preferences are personal choices about the appearance and operation of AppScan Source for Analysis.	

Scan menu

From the **Scan** menu, you manage scans of a selected application, project, or file.

Table 25. Scan menu

Menu item	Description	Keyboard shortcut
Scan All	Scan all applications. The scan will run with the default scan configuration.	
Scan Selection	Scan the selected application, project, or file. The scan will run with the default scan configuration.	F4
Scan Again	Re-scan assessment targets. The last scan configuration that was use to scan the item (or selected items) will be used again for the scan.	
Cancel Scan	Terminates the scan and does not produce any results.	
Stop Scan	Halts the scan and produces partial results.	
Build Configuration	The configuration defines the project build parameters, such as preprocessor definitions or include paths. Typically, a configuration named **Release** or **Debug** appears from an imported project. This menu item is disabled when it is not applicable.	

Tools menu

This menu includes options for comparing assessments and generating reports - and for reviewing files or findings in an editor. Some **Tools** menu items are context-sensitive and depend on the active view and the currently-selected item in that view.

Table 26. Tools menu

Menu item	Description
Diff Assessments	This action opens a dialog box that allows you to select two assessments to compare.
Generate Findings Report	Generate a report of selected findings or bundle contents. A findings or bundle view must be selected when issuing this action. If findings are not selected in the view, the report will contain all findings in the view.
Generate Report	Generate a report that allows you to view all findings based on specific compliance requirements or guidelines.

Table 26. Tools menu (continued)

Menu item	Description
Open in Internal Editor	Open a file in the internal AppScan Source for Analysis editor. This action can be used for a selected finding and will cause the file associated with the finding to open in the editor.
Open in External Editor	Open a file using an external editor. This action can be used for a selected finding and will cause the file associated with the finding to open in the editor.

Admin menu

The **Admin** menu provides actions that allow you to manage users and launch audit information.

Table 27. Admin menu

Menu item	Description
Manage Users	This action launches a dialog box that allows you to create and edit users and permissions . You must have AppScan Source administrative permissions to be able to manage users.
Change Password	This action launches a dialog box that allows you to change your password.
Audit	This action launches a view that allows you to see audit information, such as authentication events.

Refer to the *AppScan Source Installation and Administration Guide* for further details about administrative tasks.

View menu

The **View** menu controls the display of each view or selects an open view.

To learn more about the views that are available in AppScan Source for Analysis, see AppScan Source for Analysis Views.

Perspective menu

The **Perspective** menu controls the display of AppScan Source for Analysis *perspectives*, which are pre-configured collections of views and options.

Table 28. Perspective menu

Menu item	Description	Keyboard shortcut
Configuration	This perspective allows you to create and manage applications, projects, and attributes.	Alt+1

Table 28. Perspective menu (continued)

Menu item	Description	Keyboard shortcut
Triage	This perspective allows you to view scan results to prioritize remediation workflow and separate real vulnerabilities from potential ones. This perspective can be used to isolate the issues that you need to fix first.	Alt+2
Analysis	This perspective allows you to drill down into individual findings - and review source code, remediation advice, and AppScan Source trace information.	Alt+3
Reset Perspective	Selecting this causes the currently-displayed perspective to return to its default views and layouts.	

Help menu

The **Help** menu includes actions that open a variety of tools that assist with product usage. These include the product welcome, online user assistance, and the AppScan Source Security Knowledgebase.

Table 29. Help menu

Menu item	Description
Welcome	Selecting this causes the AppScan Source for Analysis Welcome view to open. This view offers quick links to a variety of help resources, including an X-Force RSS feed.
Help Contents	Selecting this causes the AppScan Source for Analysis product user assistance to open.
Security Knowledgebase	This action causes the AppScan Source Security Knowledgebase to open. The Knowledgebase provided intelligence on each vulnerability - offering precise descriptions about the root cause, severity of risk, and actionable remediation advice.
Logs	Selecting this causes the Logs view to open. Within the view, tabs allow you to select the log file to display.
About IBM Security AppScan Source for Analysis	Selecting this opens a dialog box that provides product information about AppScan Source for Analysis.

Toolbars

Toolbars in the AppScan Source for Analysis workbench provide graphical shortcuts to commands. To identify a particular toolbar icon, pause the mouse briefly over the icon until hover help appears. The toolbar buttons represent frequently used operations (also found in the **Main** menu). Toolbar operations are context-dependent.

The main toolbar provides quick links to AppScan Source for Analysis perspectives. In addition, most views have toolbars that offer a quick way of launching common actions related to the view.

Hover help

Hover help is a form of context-sensitive help that displays in a small pop-up window when the mouse pointer is over an element of the interface. A brief description of the interface element is displayed in the pop-up window.

In addition to providing hover help for buttons and icons, AppScan Source for Analysis offers hover help in a variety of places, such as:

- In the Explorer view, hover help is available to indicate the file name and path of applications, projects, and files. Hover help also indicates if an application or project is registered.
- In the Trace view, hovering over trace nodes in the graph provides information about the node.
- In the Filter Editor view **Trace** section, hovering over a trace entry provides details about it.
- In the Scan Configuration view **Advanced Settings** section, hover help is available for each setting.
- Hovering over a graph bar in the Assessment Summary view provides the exact number of findings represented by the bar.
- In the workbench status bar (located along the bottom of the workbench), hovering over the user icon launches hover help that identifies the logged on user. Hovering over the server icon launches hover help that indicates the Enterprise Server that AppScan Source for Analysis is connect to.

Status bar

The status bar, located along the bottom of the workbench, displays informational messages that identify the current action, such as a scan.

For example, during a scan, the status bar might display `Scanning <Project name>` - with a progress indicator. In addition, the current stage of the scan is displayed - for example, `Preparing for Vulnerability Analysis: 99%`. After the scan is complete, the elapsed time is display in the status bar.

The status bar also includes information about the current user and server connection. Hovering over the user icon launches hover help that identifies the logged on user. Hovering over the server icon launches hover help that indicates the Enterprise Server that AppScan Source for Analysis is connect to.

Chapter 13. Views

The AppScan Source for Analysis work environment consists of multiple perspectives and views, which contain different assessment or scan data.

AppScan Source for Analysis views provide alternative presentations of findings (some of which support code editing) - and they allow you to navigate the information in your workbench. For example, the Explorer view displays applications, projects, and other resources. A view might appear by itself, or stacked with other views in a tabbed notebook. You can change the layout of a perspective by opening and closing views and by docking them in different positions in the workbench window.

Views are described in greater detail in these sections:
- "Configuration views"
- "Views that assist with scan output" on page 236
- "Views that assist with triage" on page 240
- "Views that allow you to investigate a single finding" on page 250
- "Views that allow you to work with assessments" on page 254
- "Bundles view" on page 257

Configuration views

The views in this section are used for configuring AppScan Source.
- "Custom Rules view"
- "Explorer view" on page 57
- "Scan Rule Library view" on page 191
- "Properties view" on page 223
- "Scan Configuration view" on page 84
- "Report Editor" on page 175

Custom Rules view

In the Custom Rules view, you create custom rules with the Custom Rules Wizard. Add, view, or delete existing rules.

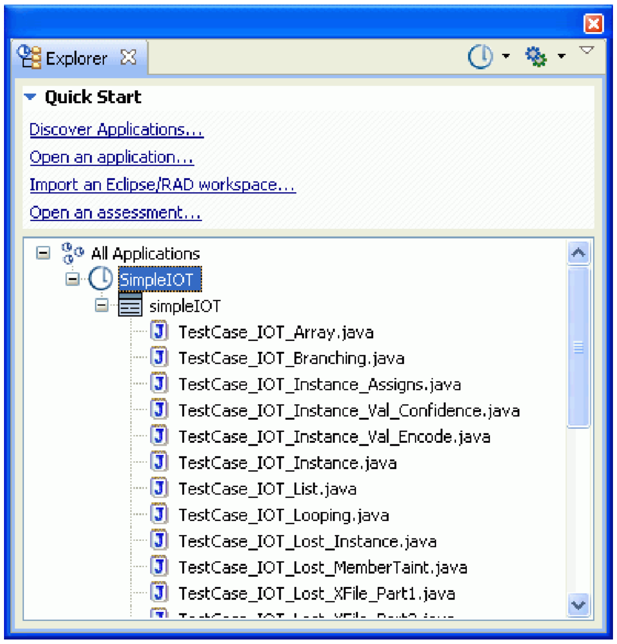
See "Creating custom rules" on page 182 for more details.

Explorer view

The Explorer view contains a **Quick Start** section at the top - and an explorer section at the bottom which contains one node, **All Applications**. The **Quick Start** section contains several useful links that launch common actions. The explorer section consists of a tree pane that provides a hierarchical view of your resources: applications, projects, directories, and project files, with **All Applications** as its root. You navigate these resources much like a file browser. As you navigate the view, the selection state of the tree determines the available tabs in the Properties view.
- "General information" on page 58
- "Quick Start section" on page 58

- "Toolbar buttons" on page 59
- "Right-click menu options" on page 59
- "Application and project indicators" on page 62

General information


Explorer
Quick Start
Discover Applications...
Open an application...
Import an Eclipse/RAD workspace...
Open an assessment...
All Applications
SimpleIOT
simpleIOT
TestCase_IOT_Array.java
TestCase_IOT_Branching.java
TestCase_IOT_Instance_Assigns.java
TestCase_IOT_Instance_Val_Confidence.java
TestCase_IOT_Instance_Val_Encode.java
TestCase_IOT_Instance.java
TestCase_IOT_List.java
TestCase_IOT_Looping.java
TestCase_IOT_Lost_Instance.java
TestCase_IOT_Lost_MemberTaint.java
TestCase_IOT_Lost_XFile_Part1.java

In the Explorer view, you add applications and projects and scan code using toolbar buttons, links in the **Quick Start** section, or right-click menu commands in the explorer section. Once you have added applications, the explorer section provides visual indicators of your applications and projects and the status of each.

Tip: In the Explorer view, hover help is available to indicate the file name and path of applications, projects, and files. Hover help also indicates if an application or project is registered.

Quick Start section

The **Quick Start** section offers these links for launching common tasks:

- **Discover applications**: This launches the Application Discovery Assistant, which allows to you to quickly create and configure applications and projects for Java and Microsoft Visual Studio source code.
- **Open an application**: This launches an Open dialog box, which allows you to browse for and add an existing application to the set of applications. File types that can be added include `.paf`, `.sln`, `.dsw`, and `.ewf`.
- **Import an Eclipse/RAD workspace**: This launches the Add Workspace dialog box, which allows you to add an existing Eclipse or IBM Rational Application Developer for WebSphere Software (RAD) workspace that contains Java projects. After the workspace has been imported, you will be able to scan any Java projects that it contains.

  Note: Before importing a workspace, be certain that you have installed and updated the development environment as described in "Configuring your

development environment for Eclipse and Rational Application Developer for WebSphere Software (RAD) projects" on page 26.
- **Open an assessment**: This launches an Open dialog box, which allows you to browse for an AppScan Source assessment file. File types that can be opened include `.ozasmt` and `.xml`.

Toolbar buttons

Table 30. Toolbar buttons

Action	Icon	Description
Add Application Menu	🕓	Clicking the down-arrow on the **Add Application Menu** button allows you to select actions for creating a new application, opening an existing application, importing a workspace, or launching the Application Discovery Assistant.
Scan Selection	⚙	The **Scan Selection** button allows you to scan the object that is selected in the explorer section. The default scan configuration will be used for the scan. To choose a different scan configuration to use for the scan, click the down-arrow on the **Scan Selection** button. Select the scan configuration that you want to use - or choose the **Edit Configurations** action to set a different scan configuration as default (in the Scan Configuration view, select the configuration that you want to set as default, and then click **Select as Default**).
View Menu		The **View Menu** button opens a menu that allows you to refresh the explorer section and hide registered items.

Right-click menu options

The availability of right-click menu options is determined by the item that is selected in the explorer section.

- When **All Applications** is selected in the explorer section, these right-click menu options are available:
  - **Scan All Applications**: Scan all applications. The scan will run with the default scan configuration.
  - **Scan All Applications With**: Select the scan configuration that you want to use - or choose the **Edit Configurations** action to set a different scan

configuration as default (in the Scan Configuration view, select the configuration that you want to set as default, and then click **Select as Default**).

- **Add Application**
  - **Create a new application**: Add a new application to the set of applications. This action launches the New Application Wizard.
  - **Open an existing application**: This launches an Open dialog box, which allows you to browse for and add an existing application to the set of applications. File types that can be added include `.paf`, `.sln`, `.dsw`, and `.ewf`.
  - **Import an existing Eclipse/ RAD workspace**: This launches the Add Workspace dialog box, which allows you to add an existing Eclipse or IBM Rational Application Developer for WebSphere Software (RAD) workspace that contains Java projects. After the workspace has been imported, you will be able to scan any Java projects that it contains.

    Note: Before importing a workspace, be certain that you have installed and updated the development environment as described in "Configuring your development environment for Eclipse and Rational Application Developer for WebSphere Software (RAD) projects" on page 26.
  - **Discover applications**: This launches the Application Discovery Assistant, which allows to you to quickly create and configure applications and projects for Java and Microsoft Visual Studio source code.
- **Expand All**
- **Collapse All**
- **Properties**: Selecting this opens the Properties view for the selected item.

- When an application is selected in the explorer section, these right-click menu options are available:
  - **Scan Application**: Scan the selected application, project, or file. The scan will run with the default scan configuration.
  - **Scan Application With**: Select the scan configuration that you want to use - or choose the **Edit Configurations** action to set a different scan configuration as default (in the Scan Configuration view, select the configuration that you want to set as default, and then click **Select as Default**).
  - **Add Project**
    - **New Project**: If an application is selected in the Explorer view, this action is available and choosing it allows you to add a new project to the application. This action launches the New Project Wizard.
    - **Existing Project**: If an application is selected in the Explorer view, this action is available and choosing it allows you to add an existing project to the application. This action launches a dialog box that allows you to browse for a `.ppf` , `.vcproj`, `.vcxproj`, `.csproj`, `.vbproj`, `.dsp`, or `.epf` file to open.
    - **Multiple Projects**: Add multiple projects to the application that is selected in the Explorer view. This action launches a dialog box that allows you to complete one of these tasks:
      - Specify a directory in which to search for projects.
      - Specify a workspace in which to search for projects.
      - Specify a Microsoft Solution file in which to search for projects.

      In the search results, you can select one or more projects to add.

- **Remove Application**: If an application is selected in the Explorer view, this action is available and choosing it removes the selected application.
  - **Add Custom Finding**: This action launches the Create Custom Finding dialog box, allowing you to create a custom finding for the selected application.
  - **Refresh**: Refresh the contents of a selected application, project, or view.
  - Register/unregister:
    - **Register Application**: Register the selected application or project with AppScan Source. You must register applications and projects before they can be published to the AppScan Source Database.
    - **Register Application As...**: Select this to reregister an application with a new name.
    - **Unregister Application**: Unregister the selected application or project.
    - **Locate**: Select this to associate a local application or project with one that has been registered by another AppScan Source user.
  - **Expand All**
  - **Collapse All**
  - **Properties**: Selecting this opens the Properties view for the selected item.
- When a project is selected in the explorer section, these right-click menu options are available:
  - **Scan Project**: Scan the selected application, project, or file. The scan will run with the default scan configuration.
  - **Scan Project With**: Select the scan configuration that you want to use - or choose the **Edit Configurations** action to set a different scan configuration as default (in the Scan Configuration view, select the configuration that you want to set as default, and then click **Select as Default**).
  - **Copy Project**: If a project is selected in the Explorer view, this action is available and choosing it opens a dialog box that allows you to copy the project to another application - or create a copy of the project in the application that currently contains the project.
  - **Remove Project**: Remove the selected object.
  - Register/unregister:
    - **Register Project**: Register the selected application or project with AppScan Source. You must register applications and projects before they can be published to the AppScan Source Database.
    - **Unregister Project**: Unregister the selected application or project.
    - **Locate**: Select this to associate a local application or project with one that has been registered by another AppScan Source user.
  - **Expand All**
  - **Collapse All**
  - **Properties**: Selecting this opens the Properties view for the selected item.
- When a file is selected in the explorer section, these right-click menu options are available:
  - **Scan File**: Scan the selected application, project, or file. The scan will run with the default scan configuration.
  - **Scan File With**: Select the scan configuration that you want to use - or choose the **Edit Configurations** action to set a different scan configuration as default (in the Scan Configuration view, select the configuration that you want to set as default, and then click **Select as Default**).
  - **Exclude from Scans**: Remove the selected file from scans.

- **Open in Internal Editor**: Open the selected file in the AppScan Source editor (in the Analysis perspective).
- **Open in External Editor**: Choose an external editor in which to open the selected file.
- **Properties**: Selecting this opens the Properties view for the selected item.

Application and project indicators

This table identifies the application and project icons in the Explorer view.

Table 31. Application and Project Icons

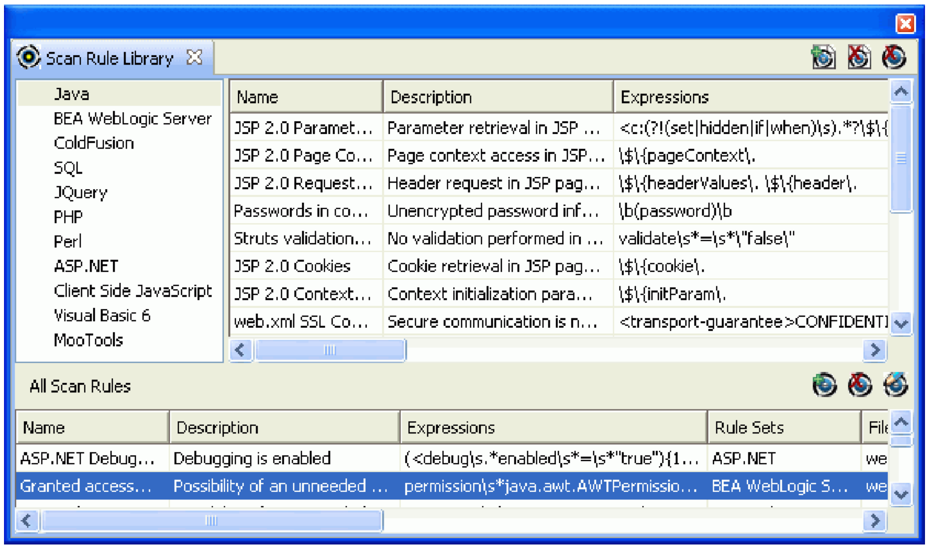
Application or project type	**Not registered**	**Registered**	**Missing/Not Found**
Imported application	◷	◷	✗
Application that is created manually or created using the Application Discovery Assistant	◷	◷	✗
Imported project	☰	☰	✗
Project that is created manually or created using the Application Discovery Assistant	☰	☰	✗

The Explorer view displays local applications and projects as well as those registered on the server (those that are registered on the server but not saved locally - for example, applications and projects registered by other users - appear greyed out). If you click the toolbar **View Menu** button and toggle the **Hide items registered on the server** menu item so it is not selected, you can view existing server applications and projects. If a project is greyed out, you can right-click and choose **Locate** in the menu.

Scan Rule Library view

Pattern-based scanning is an analysis of your source code based on customized search criteria. The Scan Rule Library view allows you to view existing pattern-based scan rules, by language (including the out-of-the-box AppScan Source scan rule library). In addition, the view allows you to add rules and patterns for pattern-based scanning.

Once you build a scan rule library, you can apply the pattern analysis to specific applications or projects. See "Customizing with pattern-based scan rules" on page 186 for details about pattern searches.



Java
BEA WebLogic Server
ColdFusion
SQL
JQuery
PHP
Perl
ASP.NET
Client Side JavaScript
Visual Basic 6
MooTools

Name	Description	Expressions
JSP 2.0 Paramet...	Parameter retrieval in JSP ...	<c:(?!(set|hidden|if|when)\s).*?\$\{
JSP 2.0 Page Co...	Page context access in JSP...	\$\{pageContext\.
JSP 2.0 Request...	Header request in JSP pag...	\$\{headerValues\. \$\{header\.
Passwords in co...	Unencrypted password inf...	\b(password)\b
Struts validation...	No validation performed in ...	validate\s*=\s*\"false\"
JSP 2.0 Cookies	Cookie retrieval in JSP pag...	\$\{cookie\.
JSP 2.0 Context...	Context initialization para...	\$\{initParam\.
web.xml SSL Co...	Secure communication is n...	<transport-guarantee>CONFIDENTI

All Scan Rules

Name	Description	Expressions	Rule Sets	File
ASP.NET Debug...	Debugging is enabled	(<debug\s.*enabled\s*=\s*"true"){1...	ASP.NET	we
Granted access...	Possibility of an unneeded ...	permission\s*java.awt.AWTPermissio...	BEA WebLogic S...	we

Properties view

The contents of the Properties view depend on the item that is selected in the Explorer view. Properties apply to all applications, individual applications, projects, or files. Visible properties depend on the language or selected project type.

- "Properties view: all applications"
- "Properties view: selected application" on page 193
- "Properties view: selected project" on page 195
- "File properties" on page 231

Properties view: all applications

If you select **All Applications** in the Explorer view, the Properties view displays the Overview and Filters tabs.

Overview

The Overview tab displays global attributes. *Attributes* are named groupings of user-defined items with similar characteristics. You add or delete attributes and their values.

Filters

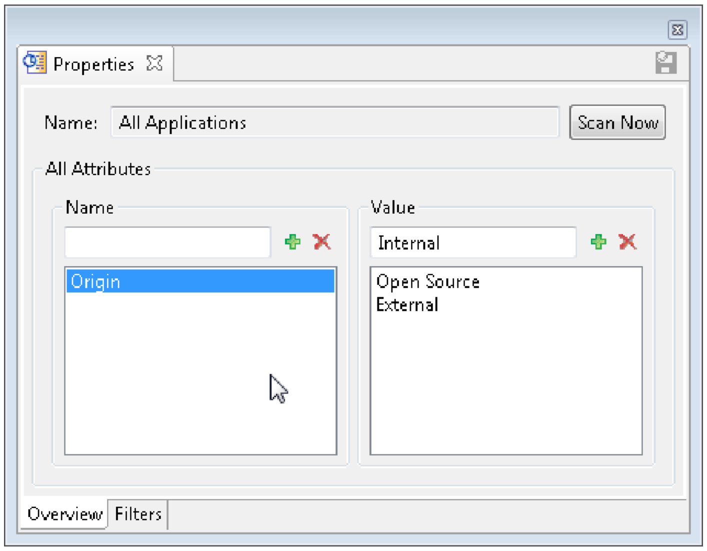
This tab allows you to specify existing filters for all applications, and how you want the filters applied (a filter can be applied directly - or its inverse can be applied). See Chapter 5, "Triage and analysis," on page 99 for information about filters - and "Applying filters globally" on page 118 for details about applying them globally.

Filtered findings do not appear in scan results or factor into application or project metrics.

Adding and removing global attributes:

You must define attributes for **All Applications** before grouping attributes for applications.

About this task



To delete a global attribute or its value, select the attribute name or attribute value and click **Remove Attribute**. The name or value no longer appears in the list.

Note: Deleting an attribute does not affect historical results.

To add a global attribute and its value, follow the steps below.

Procedure

1. Select **All Applications**.
2. In the Properties view Overview tab, type a name for the attribute.
3. Click **Add Attribute**. The attribute name appears in the Name list.
4. Select the named attribute.
5. Type a **Value** for the attribute.
6. Click **Add Value**. The attribute value appears in the value list.

Properties view: selected application

In this view, you configure attributes for the selected application. Application attributes depend on previously-created global attributes.

- “Overview” on page 193
- “Exclusions and Filters” on page 194
- “Scan Rules and Rule Sets” on page 194
- “Modified Findings” on page 194
- “Custom Findings” on page 195

Overview

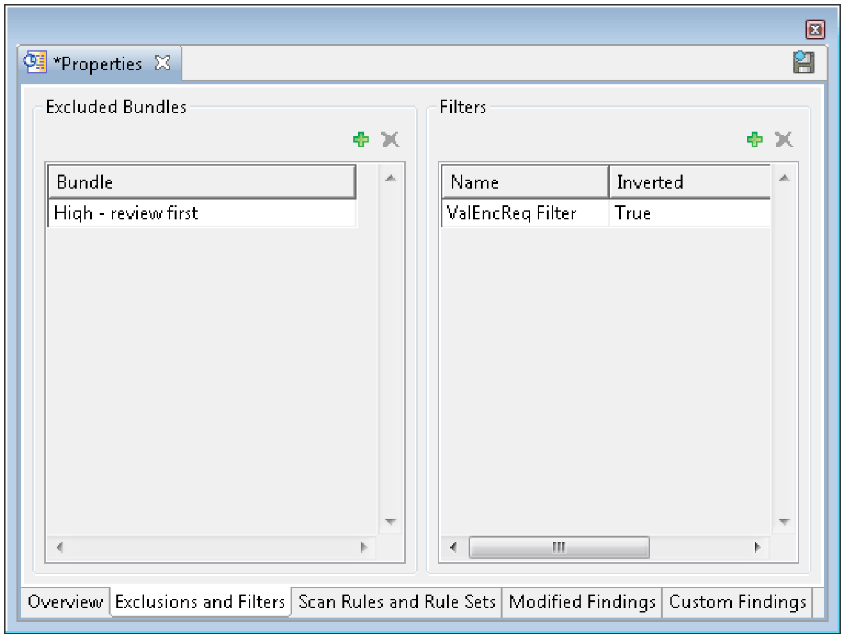
The Overview tab displays:

- The application name. The application can be renamed by entering a new name in the field.
- Application attributes

Exclusions and Filters

This tab allows you to specify existing filters for the selected application, and how you want the filters applied (a filter can be applied directly - or its inverse can be applied). In the tab, you can also manage bundles that exclude results from your scan. See Chapter 5, “Triage and analysis,” on page 99 for information about filters - and “Applying filters globally” on page 118 for details about applying them globally.

Excluded and filtered findings do not appear in scan results or factor into application or project metrics.



Bundle
High - review first

Name	Inverted
ValEncReq Filter	True

Scan Rules and Rule Sets

When you select an application in the Explorer view, the Scan Rules and Rule Sets tab in the Properties view allows you to manage application scan rules. Using pattern-based scanning, you search for text patterns that you want to appear as findings. When you create a scan rule set, you can apply it to an application or to a project (individual scan rules can only be applied to projects). See “Customizing with pattern-based scan rules” on page 186 for details about pattern-based analysis.

Modified Findings

On the Modified Findings tab, you view, edit, or delete any previously modified findings, or modify an existing finding. *Modified findings* are findings with altered vulnerability type, severity, classification, or notes.

Custom Findings

On the Custom Findings tab, you view, add, edit, or delete custom findings. See “Custom findings” on page 132 for more details.

Creating an application attribute:

Procedure

1. On the Overview tab, click **Add Attributes**.
2. In the **Global Attributes** dialog box, select the name of the attribute to apply to the application.
3. Click the **Value** column and select the attribute value from the list.

Properties view: selected project

In this mode of the Properties view, you configure parameters for the selected project. Project attributes depend on previously created global attributes. Properties vary according to the selected project.

- "Selected project Overview tab" on page 195
- "Filters" on page 196
- "Scan Rules and Rule Sets" on page 196
- "File Extensions" on page 196
- "Sources" on page 197
- "JavaServer Page (JSP) Project Dependencies" on page 198
- "Project Dependencies" on page 198
- "Compilation" on page 198
- "Optimizations" on page 199
- "Precompile Tab (ASP.NET only)" on page 199

Selected project Overview tab

The Overview tab displays:

- Project **Name**. The project can be renamed by entering a new name in the field.
- Project **File** name and path
- **Project Type**
- Configuration: This section displays the target configuration. For .NET and C++ projects, this section displays the target configuration that has been saved in the Project Dependencies tab. For all other project types, this section displays **Default**.
- Filter option: Select **Filter findings contained in external sources** to filter out any findings discovered in files that are not source files of the scanned project. This option reduces noise for projects where findings are reported in compiler-generated or temporary files, such as `ASP.NET`.
- Vulnerability analysis cache options: If you are refining your assessment of a code base by scanning iteratively and adding custom rules, and then re-scanning without changing the source code, you can dramatically reduce scan time by setting the project properties to use a vulnerability analysis cache. To do this, select the **Enable Vulnerability Analysis cache** check box in the project properties. The first time you scan the project after selecting this check box, a vulnerability analysis cache will be created. For every subsequent scan of the project, the vulnerability analysis cache will be used and scan time will be reduced.

  To clear the vulnerability analysis cache, click **Clear cache**. The next time you scan the project, a new vulnerability analysis cache will be created (provided that you do not deselect the **Enable Vulnerability Analysis cache** setting). You may want to clear the cache if:

  – Source code in the project has changed since the last scan.

- You have made project configuration changes, such as adding or deleting source files.
- You have changed code configuration options. For example, you may want to clear the cache if you are scanning Java and the class path has changed - or if you are scanning C or C++ and you have changed the `include` path or preprocessor definitions.

Note: You can also clear the vulnerability analysis cache by selecting the **Clear cache** check box when creating custom rules in the Custom Rules Wizard.

- **String Analysis**: String analysis monitors string manipulation in Java or Microsoft .NET projects. It provides the automatic detection of sanitizer and validator routines. With this detection, false positives and negatives can be reduced. Select the **Enable String Analysis to find validator/sanitizer functions** check box to enable string analysis. The **Apply imported rules to Global Scope** check box determines if the discovered sanitizer or validator routines should be applied to a single project or on a global level (to all projects).

  Note: The application of string analysis can slow a scan. It is therefore recommended that it should only be applied after code changes and then disabled for subsequent scans. In addition, the discovered routines should be viewed as *suggestions* and reviewed by auditors. These routines can be viewed in the Custom Rules view.
- **File Encoding**: The character encoding of files in your project must be set so that AppScan Source can read the files properly (and, for example, display them correctly in the source view).

  Note: The default file encoding for AppScan Source projects is **ISO-8859-1**. The default file encoding can be changed in the General preference page.

Filters

This tab allows you to specify existing filters for the selected project, and how you want the filters applied (a filter can be applied directly - or its inverse can be applied). See Chapter 5, “Triage and analysis,” on page 99 for information about filters - and “Applying filters globally” on page 118 for details about applying them globally.

Scan Rules and Rule Sets

When you select a project in the Explorer view, the Scan Rules and Rule Sets tab in the Properties view allows you to manage application scan rules. Using pattern-based scanning, you search for text patterns that you want to appear as findings. When you create a scan rule, you can apply it to an application or to a project. See “Customizing with pattern-based scan rules” on page 186 for details about pattern-based analysis.

File Extensions

Use this tab to configure or add valid file extensions for the project - and to exclude files from scans and specify extensions as web files.

The **File Extensions** section lists the extensions that have been set globally in the “Project file extensions” on page 76 preference page for the current project type (you can choose file extensions for a different project type using the **File Extension**

Set menu). To exclude an extension from scans of the current project, select it in the list and click **Exclude Extension**. This causes the extension to be listed in the **Excluded Extensions** section of the tab.

To add an additional extension for the project, select **Add Extension** in the **Additional Extensions** section, and then enter the file extension and indicate if files with the extension should be scanned, considered as a web file, or excluded.

Table 32. File extension settings

Setting	Description	Usage examples
Scan or **Assess**	Include files with the indicated extension in full analysis.	• If a `.xxx` extension is created for Java projects and marked as **Scan** or **Assess**, files with that extension will be compiled and scanned. • A file can be part of a project but not marked as **Scan** or **Assess** if it should not be compiled and scanned (such as header files in C++). These files would be included in the project and searched during pattern-based analysis.
Web File	Mark files with the indicated extension for JSP compilation. This setting allows AppScan Source to separate web sources from non-web sources.	If a `.yyy` extension is created for Java projects and marked as a **Web File**, files with that extension will be arranged as web sources in the projects. When AppScan Source prepares for analysis, these files will be precompiled into classes to be analyzed.
Exclude	Do not create source files in the project for files with the indicated extension. Files with this extension will not be scanned.	Create a `.zzz` extension for files that are necessary to your projects for compilation, but do not need to be included in analysis.

Sources

Specify the sources to include in the scan.

- Working Directory: The location of the AppScan Source project file (`ppf`) and the base for all relative paths.
- **Add Source Root** and **Remove Source Root**: The Sources tab displays the properties established for the project from the Project Configuration Wizard or defined in the imported `ppf`.

  Remove Source Root is available only when the **Source Root** icon is selected. It is used to remove the source code root directory.
- Find Source Roots (Java projects only): Allow AppScan Source for Analysis to find all valid source roots automatically.
- Project files are listed under the **Source Root** icon. Files that are excluded from scanning have a red file icon (if an excluded file is right-clicked, its menu has

Exclude disabled and **Include** enabled). To exclude an included file, right-click it and choose **Exclude** in the menu. To include an excluded file, right-click it and choose **Include** in the menu.

JavaServer Page (JSP) Project Dependencies

The JSP Project Dependencies tab displays the properties established for the specified JSP project.

- Contains web (JSP) content: Identifies if the project is a web application that contains JavaServer Pages.
- Web context root: A `WAR` file or a directory that contains the `WEB-INF` directory. The web context root must be the root of a valid web application.
- JSP Compiler: Tomcat 7 (Jasper 2) is the default JSP compiler setting (the default JSP compiler can be changed in the Java and JSP preference page). To learn about the compilers that are supported by AppScan Source, see http://www.ibm.com/support/docview.wss?uid=swg27027486.

  Apache Tomcat Versions 5, 6, and 7 are included in the installation of AppScan Source. If the **Tomcat 5**, **Tomcat 6**, and **Tomcat 7** preference pages are not configured, AppScan Source will compile JSP files using the supplied Tomcat JSP compiler that is currently marked as default. If you want to employ an external Tomcat compiler, use the Tomcat preference pages to point to your local Tomcat installation.

  If you are using Oracle WebLogic Server or WebSphere Application Server, you must configure the applicable preference page to point to your local installation of the application server so that it can be used for JSP compilation during analysis. If you have not already completed this configuration, you will be prompted by a message to do so when you select the JSP compiler. If you click **Yes** in the message, you will be taken to the appropriate preference page. If you click **No**, a warning link will display next to the JSP compiler selection (following the link will open the preference page).

Project Dependencies

The Project Dependencies tab displays project properties. Configuration settings on this tab vary by language, for example:

- **Options** allow you to select any additional required compiler parameters.
- JDK settings are specific to Java.
- Preprocessor definitions are specific to C/C++ code.
- The target configuration is available only for .NET and C++ projects.

Compilation

- Options: Additional required compiler parameters for the project configuration.
- Use JDK: Identify the JDK used for the project compilation, as configured in Preferences. See Chapter 3, “Preferences,” on page 63.

  Java projects may refer to a local Java Development Kit (JDK) location. When projects move to the server, the JDK path may no longer be valid. To transfer local projects to the server, you must identify the default JDK path for each project that specifies a named JDK.

  Note: The default compiler for JSP projects is Tomcat 7 (Jasper 2), which requires Java Version 1.6 or higher. If **Tomcat 7** is kept as default, selecting an earlier JDK (for example, **IBM JDK 1.5**) will result in compilation errors during scans.

- Validate: **Validate** assures that project dependencies are correctly configured. It checks Java projects for configuration conflicts between sources and the class path and for compilation errors. **A conflict exists** if a class in a class path is duplicated in a source root. (If a conflict exists, modify the class path to remove the conflicting class.)

  After checking for conflicts, **Validate** determines if the project compiles and reports any compilation errors.

Optimizations

- **Precompiled classes**: Use precompiled class files instead of compiling during the scan. When selected, this option disables the source stage options.
- Stage sources files to minimize effects of compile errors: Controls whether AppScan Source copies the sources to the staging directory.

  Correct for packages not matching the directory requires Java Compile to open each source file.

  Clean staging area between scans increases performance between scans.

Precompile Tab (ASP.NET only)

Precompilation is accomplished by making an HTTP request to a special page (by default, `precompile.axd`) in the website. This page is processed by a special HTTP handler specified in the `web.config`. This handler compiles the entire site, including `client.aspx` files, to the Temporary ASP.NET Files directory in the .NET framework directory, where they are all then scanned.

To scan ASP.NET 1.1, you must instrument the website such that the website compiles and builds debug information. Subsequently, the fact that a website compiles and builds debug information is, in and of itself, a security vulnerability. You can safely ignore this vulnerability as the scan requires it. However, ensure that your deployed application is not compiled with `debug=true` in the `web.config`.

To precompile an ASP.NET 1.1 website, add this element as a child to the `<system.web>` element in your `web.config` file:

```
<httpHandlers><add verb="*" path="precompile.axd"
type="System.Web.Handlers.BatchHandler"/></httpHandlers>
```

You should also set `debug=true` in the compilation element. For example:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
 <system.web>
  <httpHandlers><add verb="*" path="precompile.axd"
  type="System.Web.Handlers.BatchHandler"/>
  </httpHandlers>
   <compilation
    defaultLanguage="c#"
    debug="true"
   />
...
```

This element specifies to the website that the page, `precompile.axd` will be handled by the special .Net `System.Web.Handlers.BatchHandler` class. This class precompiles the contents of the website to the Temporary ASP.NET Files directory.

- Website: Request target to precompile the site. The default location is `precompile.axd`. `Precompile.axd` is a virtual file and maps to the file specified in the `web.config` file.

- Output directory: The directory targeted by the precompilation. AppScan Source finds the precompilation output in this directory.
- Precompile the ASP.NET website: AppScan Source automatically precompiles and scans the precompiled output during a scan.
- Stop scan if precompile fails: Selecting **Precompile the ASP.NET website** and **Stop scan if precompile fails**, stops a scan if precompilation fails. Otherwise, the scan continues only on the website's primary output.
- Compile Now: Test to see that the precompilation, based on the current settings, succeeds before scanning. Compilation output displays in the Precompile Output pane.
- Additional Assemblies: For any .NET project type, specify additional assemblies to scan.
- Project References: List the directories in which to search for referenced assemblies in .NET assembly projects and existing .NET projects.

File properties

File properties are similar to project dependencies, commonly configured for C/C++ applications.

Include configuration data from project: Include the project configuration data in the file configuration. The file configuration then consists of the cumulative project configuration and file configuration. File configuration supersedes project configuration.

Scan Configuration view

The Scan Configuration view allows you to create configurations that you can use when launching scans - and you can set the default scan configuration in the view. In a scan configuration, you can specify source rules to use during a scan - and you can include numerous scan settings. The settings made in a scan configuration can often lead to better scan results - and the ability to save these settings can make scanning easier and more time-efficient.

The Scan Configuration view has four main sections:
- “Scan configuration management” on page 84
- “General” on page 85
- “Scan Rules” on page 85
- “Advanced Settings” on page 86

Note: Scan configurations do not apply to quality scans or when scanning JavaScript, ColdFusion, Perl, Cobol, PL/SQL, or T-SQL.

Scan configuration management

Use this section to select, add, remove, save, and share scan configurations - and to set scan configurations as default.
- To create a new scan configuration, click **New**. After completing the scan configuration settings, click **Save** to save the changes. To set the scan configuration as default, click **Select as Default** after saving it. To learn how the default scan configuration is used, refer to “Scanning source code” on page 79.
- To work with an existing scan configuration, select it from the list:
  - If you modify the scan configuration settings, click **Save** to save the changes (unwanted changes can be discarded by switching to a different scan configuration and then clicking **Discard**).

- To remove the selected scan configuration, click **Delete**.
- To duplicate the scan configuration, click **Duplicate**. This will cause a new scan configuration to be created based on the original scan configuration's settings.
- To set the scan configuration as default, click **Select as Default**. To learn how the default scan configuration is used, refer to “Scanning source code” on page 79.
- To share the scan configuration with others, click **Share**. This will save the scan configuration to the AppScan Enterprise Server.

  Note: To share scan configurations - or modify or delete a shared scan configuration - you must have must have **Manage Shared Configurations** permission. To learn about setting permissions, see the *IBM Security AppScan Source Installation and Administration Guide*.

Note: AppScan Source provides built-in scan configurations. These cannot be modified or removed. Selecting them in the list will allow you to duplicate them or view their settings.

General

This section allows you to name scan configurations and provide descriptions for them.

Scan Rules

Use this section to determine which source rules will be in effect for the scan.

A source is an input to the program, such as a file, servlet request, console input, or socket. By excluding some source rules, you can speed up scanning and avoid detecting vulnerabilities arising from inputs that are not of interest.

Rules are tagged with rule properties to indicate that they are related to a particular vulnerability, mechanism, attribute, or technology. These properties are grouped into rule sets, which correspond to a common set of related rules. You can limit the source rules included in the scan by specifying either rule sets or individual rule properties.

- Select one or more vulnerability types (organized by type in rule sets) to include in the scan:
  - **Everything**: If this is selected, vulnerabilities arising from all supported sources of input will be detected.
  - **User Input**: If this is selected, vulnerabilities arising from end user input will be detected.
  - **Web Applications**: If this is selected, vulnerabilities arising from web application risk will be detected.
  - **Error Handling and Logging**: If this is selected, vulnerabilities arising from error handling and logging mechanisms will be detected.
  - **Environment**: If this is selected, vulnerabilities arising from configuration files, system environment files, and property files will be detected.
  - **External Systems**: If this is selected, vulnerabilities arising from external entities will be detected.
  - **Data Store**: If this is selected, vulnerabilities arising from data stores (such as databases and caching) will be detected.

  - **Unusual Things**: If this is selected, vulnerabilities arising from routines that are not normally part of a production application will be detected.
  - **File System**: If this is selected, vulnerabilities arising from file systems will be detected.
  - **Sensitive Data**: If this is selected, vulnerabilities arising from sensitive data will be detected.

  Hover text describes each rule set in this section.
- Select individual scan rule properties to include in the scan: Click **Discard selected rule sets and let me select individual rule properties**. This opens the Select Rule Properties dialog box, which allows you to choose individual rule properties. If this dialog box is completed, any rule sets that were selected will be discarded. Scan rules that have the selected rule properties will be used for the scan.

Advanced Settings

This section is intended for advanced users only. It contains a variety of settings that can improve scan results. Hover text describes each setting in this section.

Report Editor

With the Report Editor, you can edit custom reports or templates or create a new report. Custom reports include any items that are available to a findings report, such as finding information, code snippets, AppScan Source trace, and remediation content, as well as a vulnerability matrix. Before you start designing new reports, it is recommended that you become familiar with the report creation process by modifying an existing report template in the Report Editor.

The Report Editor consists of the Report Layout, Categories, and Preview tabs.
- **Report Layout**: Design the appearance of the report. In the layout, you add, remove, and reorder AppScan Source report elements.
- **Categories**: Create and edit categories. A *category* is a group of findings. The category identifies the findings to include in the report, how to group those findings, and the grouping order.
- **Preview**: Look at the report for the current assessment as you edit it.

The three tabs contain common fields:
- **File**: Path of the saved grouping file (read-only). Nothing displays in this field until the file has been saved. Once saved, the grouping file is an XML file that defines the report.
- **Name**: User-defined report name.

Toolbar buttons for saving, opening, creating, copying, and generating custom reports include:
- **Create a new report**: Creates a new custom report
- **New report from existing**: Creates a new custom report from an existing report template
- **Open a saved report**: Open a grouping file to edit
- **Save**: Save the current report to the specified file
- **Save As**: Save the current report to a new file
- **Generate an instance of this report**: Create a copy of the report for the currently open assessment

Tip: To view samples of existing reports, click **New report from existing** and then choose one of the AppScan Source report templates. Exploring the Report Layout and Categories tabs in the templates will give you a feel for how reports are designed.

Report Layout tab

The Report Layout Tab consists of Palette and Layout sections - and sections that allow you to specify a header or footer that appears on each page.

Page Header and Page Footer

The **Page Header** field allows you to specify text that appears at the top of every report page - while the **Page Footer** field allows you to specify text that appears at the bottom of every page.

Palette

The Palette displays a list of elements that comprise AppScan Source standard reports. Some of the elements only display information for categories that have been defined in the Categories tab (see Table 19 on page 177).

Table 33. Report Layout Palette - elements that do not depend on categories

Report Element	Description
Text Header	Adds a bold block of text to the report layout.
Image Header	Displays an image scaled to a specified size in pixels.
AppScan Source Header	Report header containing AppScan Source branding.
Title and Date	Report title that includes name of the item that was scanned - along with the date of scan and the date the report was generated.
Text Block	Any user-defined text. A heading can also be added for the text block in the **Label** field.
Vulnerability Matrix	Assessment vulnerability matrix (displays the same graph that appears in the Vulnerability Matrix view).
Metrics	Identifies the total number of packages, classes, methods, and lines of code in all of the packages in the project.
Scan History	Metrics for the current scan and historical metrics for scans of the same target.

Table 34. Report Layout Palette - elements that depend on categories

Report Element	Description
Report Card	Brief breakdown of vulnerability levels of every category defined in the Categories tab. Contains links to the report details and severity indicators summarizing the section.
Vulnerability Breakdown	Table with a breakdown of the number of vulnerabilities in all categories defined in the Categories tab, by severity and classification.

Table 34. Report Layout Palette - elements that depend on categories (continued)

Report Element	Description
Partial Report Card	Breakdown of vulnerability levels of user-specified categories as specified in the Categories tab.
Categories	Lists all categorized findings data as defined in the Categories tab.
Category	Lists all findings in one or more categories that have been defined in the Categories tab.

Layout

As you add items from the palette, they appear in the Layout. Use the section toolbar to remove, modify, or move items in the layout.

Categories tab

Using the Categories tab, you can add categories that contain findings based on bundles, properties, or selected findings that you choose. The categories can then be used when adding certain items to the Layout. For example, when you add a Vulnerability Breakdown to the Layout, a table with a breakdown of the number of vulnerabilities in all categories (by severity and classification) is added to the layout. The Categories tab consists of a pane with a tree of categories and a pane in which to edit the attributes of the selected category. Each category contains the findings in the assessment that satisfy certain requirements that you define.

The available categories include:

- Bundle: A bundle category consists of a list of bundle names. Any finding in a bundle whose name appears in the list appears in this category. Although you choose bundles from the current assessment, you can apply the bundle category to any assessment since bundles are matched by name.
- Individual findings: Choose specific findings to add to the category. Only a snapshot of the finding is added to the report. If you modify the finding after it is added to the report, the report does not reflect the change.
- Vulnerability Types, Mechanisms, and Technologies properties: Choose sets of properties and required properties from APIs in the AppScan Source Security Knowledgebase. If a finding contains at least one of the **Properties** and all **Required Properties**, it is included in the report.

This table identifies the category panes and the items comprising the pane.

Table 35. Categories tab attributes

Attribute	Description	How to edit
Label	The brief name of the category, such as Buffer Overflow. The label identifies the category in the tree list of categories - and it is the category heading in the custom report.	Type a label in a single line text field.

Table 35. Categories tab attributes (continued)

Attribute	Description	How to edit
Summary	A template for a sentence stating how many findings are reported in this category. The actual count replaces `%FindingCount%` during report generation.	Type a short description of the category and click **Add Count** to place the variable, `%FindingCount%` in the phrase at the cursor location.
Text	Brief category description.	Enter text describing the category.
Properties (Property categories only)	Findings having at least one of these properties will be reported in this category. If a finding does not have all listed required properties, then the finding is not included in this category.	Click **Add** on the toolbar and select a property from the Add Properties dialog box. Click **Remove** to remove the selected items from the list.
Required Properties (Property categories only)	Findings with all required properties and at least one property appear in the report under this category.	Click **Add** on the toolbar and select a property from the Add Properties dialog box. Click **Remove** to remove the selected items from the list.
Bundles (Bundle categories only)	Specifies the names of bundles to include in this category.	Click **Add a bundle** in the Bundles section, and select the bundles from the list.
Findings (Findings categories only)	Specifies the findings to include in this category.	Select findings in any findings table and then click **Add findings** on the table toolbar to add the selected findings. If more than one view contains selected findings, you will be prompted to select the view that contains the selected findings that you want to add. You can also drag findings from a findings table to the table in the Report Editor view - or in the Report Editor or directly to an existing findings category in the category tree.

Preview tab

You can preview AppScan Source for Analysis reports as you edit your templates. From the Preview pane, click **Preview** to see the report for the open assessment.

Views that assist with scan output

The views in this section are used for viewing and managing scan output.

- "Console view" on page 237
- "Metrics view" on page 237
- "Quality Metrics view" on page 238

- “My Assessments view” on page 238
- “Published Assessments view” on page 239

Console view

The Console view displays output for the current scan, including status information, output text, and error messages. This view may display two consoles, one for the currently running scan and one for the completed scan.

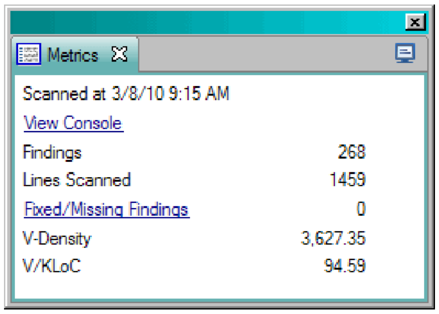
An output console displays the complete scan output, including the files scanned, the total number of files scanned, total number of vulnerabilities found, time of scan, and vulnerability density.

Toolbar buttons manipulate the console output.

An error console displays the output error messages and the number of errors in the scan. The errors value updates during the scan.

Metrics view

The Metrics view presents statistics on a per-assessment basis and includes lines of code scanned, total number of findings, V-Density, and V/KLoC.



Scanned at 3/8/10 9:15 AM
View Console

Findings	268
Lines Scanned	1459
Fixed/Missing Findings	0
V-Density	3,627.35
V/KLoC	94.59

View Console

Hyperlink that opens the Console view to see the output of the current scan.

Findings

Number of findings identified by the scan.

Lines Scanned

Number of lines of code scanned.

Fixed/Missing Findings

Number of items contained in the application bundles but not found in this scan.

V-Density

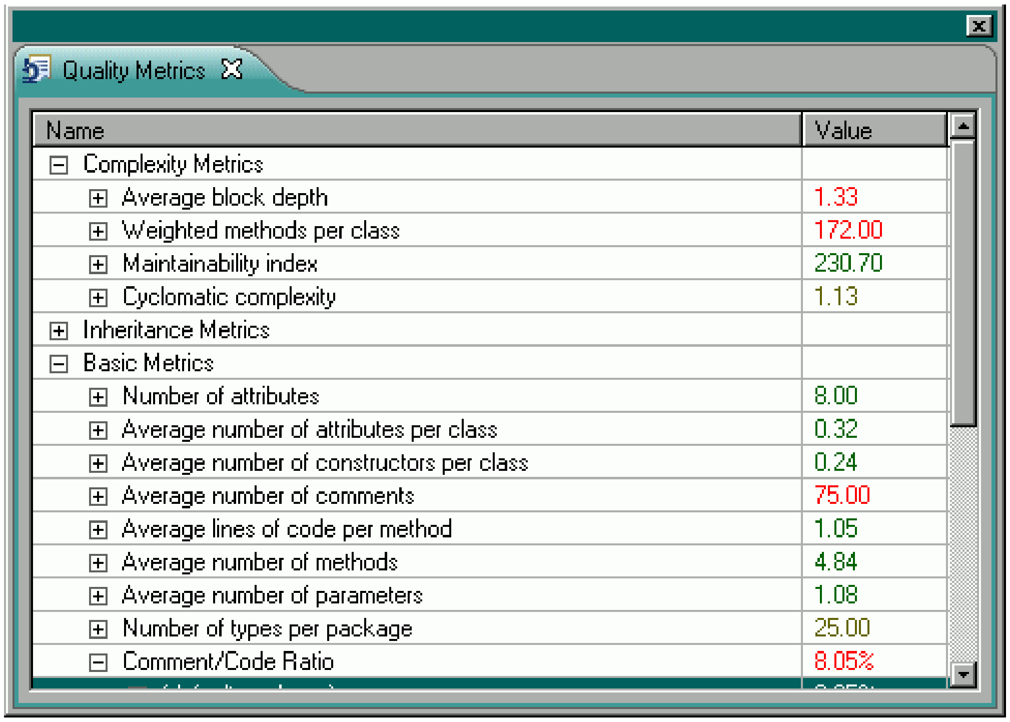
A numerical expression that enables a consistent way to evaluate the vulnerability of your applications. V-Density is calculated by relating the number and criticality of findings to the size of the application or project being analyzed.

V/KLoC

Vulnerabilities found per thousand lines of code.

Quality Metrics view

The Quality Metrics view is used for presenting the results of a Java or C/C++ software metrics quality scan. It presents a read-only compact view of analysis results organized by metric category.



Name	Value
⊟ Complexity Metrics	
⊞ Average block depth	1.33
⊞ Weighted methods per class	172.00
⊞ Maintainability index	230.70
⊞ Cyclomatic complexity	1.13
⊞ Inheritance Metrics	
⊟ Basic Metrics	
⊞ Number of attributes	8.00
⊞ Average number of attributes per class	0.32
⊞ Average number of constructors per class	0.24
⊞ Average number of comments	75.00
⊞ Average lines of code per method	1.05
⊞ Average number of methods	4.84
⊞ Average number of parameters	1.08
⊞ Number of types per package	25.00
⊟ Comment/Code Ratio	8.05%

To view metrics gathered during the scan, open a top-level metric category to list the metrics within it. For each metric category, a rolled up value is provided for all projects analyzed.

In a metric category, open a metric to display a list of packages. Each metric lists a rolled up value for one or more packages.

For a metric, open a package to display a list of classes. Each package lists a rolled up value for one or more package classes.

You can select each metric entry and press the **F1** keyboard key to see help that describes the metric.

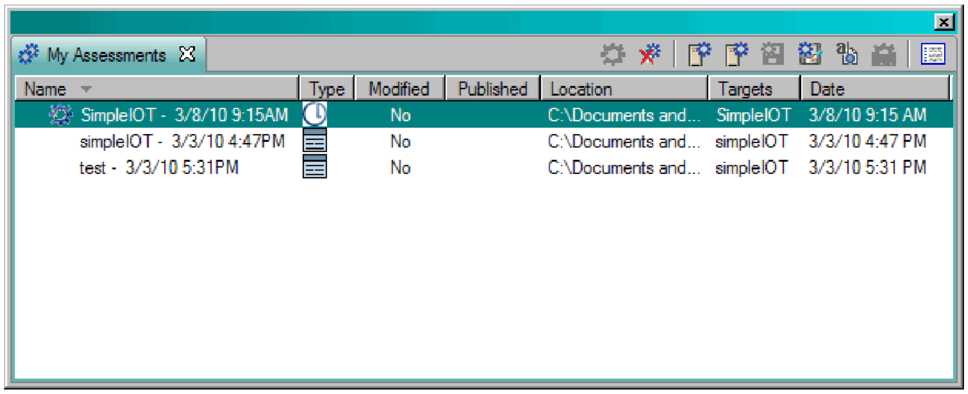
My Assessments view

The My Assessments view contains a list of assessments (the currently-opened assessment, along with any assessments that you have saved). If the current working set of assessments is modified (for example, if you add a new assessment or modify an assessment), an asterisk beside the view title indicates that there are unsaved changes in the working set.

- **Name**: Assessment name.

- **Type**: An icon indicating that the scan covered applications ( ), projects ( ), or files ( ). A star beside the assessment name indicates that the assessment is currently open.
- **Scan Configuration**: Scan configuration that was used for the scan.

  Note: Scan configurations used for quality scans do not display in this column.
- **Modified**: **Yes** or **No** indicates the modified state of the assessment.
- **Published**: Indicator that the assessments are published to the AppScan Source Database.
- **Location**: Path to the assessment file (`<file_name>.ozasmt`).
- **Targets**: Applications, projects, or files scanned.
- **Date**: Scan completion date.



Name	Type	Modified	Published	Location	Targets	Date
SimpleIOT - 3/8/10 9:15AM		No		C:\Documents and...	SimpleIOT	3/8/10 9:15 AM
simpleIOT - 3/3/10 4:47PM		No		C:\Documents and...	simpleIOT	3/3/10 4:47 PM
test - 3/3/10 5:31PM		No		C:\Documents and...	simpleIOT	3/3/10 5:31 PM

When scans complete, they automatically appear in the My Assessments view. The assessments visible in this view include scans from this computer or those that you added.

In this view, you can open, add, remove, publish, save, rename, or compare assessments. If you remove the assessment from this view without saving or publishing, it is permanently deleted. Note that every saved assessment includes all results, the output, and error logs. See "Saving assessments" on page 94 for details about saving and publishing assessments.

See "Comparing two assessments in the Assessment Diff view" on page 131 for details about comparing assessments.

Tip: A **General** preference sets the maximum number of assessments to display in the Published Assessments and My Assessments views.

Published Assessments view

The Published Assessments view lists the assessments that have been published to the AppScan Source Database.
- **Name**: Assessment name.
- **Type**: An icon indicating that the scan covered applications ( ), projects ( ), or files ( ). A star beside the assessment name indicates that the assessment is currently open.
- **Scan Configuration**: Scan configuration that was used for the scan.

  Note: Scan configurations used for quality scans do not display in this column.
- **Publisher**: User name of the person who published the assessment

- **Targets**: Applications, projects, or files scanned.
- **Date**: Scan completion date.

In the Published Assessments view, you can:
- Add assessments to the My Assessments view
- Filter assessments
- Open and delete assessments
- Close assessments
- Compare assessments
- Save assessments
- Rename assessments
- View metrics

Tip: A **General** preference sets the maximum number of assessments to display in the Published Assessments and My Assessments views.

Views that assist with triage

The views in this section are used for fine-grained scan output viewing and management.
- "Assessment Diff view"
- "Custom Findings view" on page 241
- "Excluded Findings view" on page 245
- "Findings view" on page 243
- "Fixed/Missing Findings view" on page 246
- "Modified Findings view" on page 246
- "Search Results view" on page 246
- "Report view" on page 247
- "Sources and Sinks view" on page 249

Assessment Diff view

The Assessment Diff view represents a combination of the My Assessments view and the Findings view. When you select two assessments to compare, the differences between the two assessments display.

In this view, you see the total number of new, fixed/missing, and common findings.
- Common findings appear in both assessments
- New findings are those findings that only appear in the most recent of the two assessments (blue highlight)
- Fixed/missing findings are findings that only appear in the earlier assessment (italicized and green highlight)

The right pane displays findings. Right-click a finding in the table to:
- Generate a Findings Report
- Submit the findings as defects
- Open in an external editor
- Open in the internal editor

The left pane lists assessments to compare.

Note: The Assessment Diff view ignores filters.

Custom Findings view

The Custom Findings view displays the user-defined or *custom findings* that exist in the currently opened assessment. In this view, you can create, delete, and modify custom findings for the current assessment. When a custom finding is created in the Custom Findings view, the new finding is added to the current assessment, and the assessment metrics update.

Filters and bundles do not affect the findings in the Custom Findings view. In this view, you cannot view custom report results, or save selected findings.

Views with findings

Many AppScan Source for Analysis views contain findings:

- Findings view
- Modified Findings view
- Custom Findings view
- Excluded Findings view
- Bundle views
- Fixed/Missing Findings view
- Report view
- Search Results view
- Assessment Diff view

Findings table

This table describes the columns that are available in findings tables. If a column is unavailable, it is likely hidden from the table. To select a column for viewing (or perform any other customization tasks in a table), follow the instructions in "Customizing the findings table" on page 243.

Table 36. Findings table

Column Heading	Description
Trace	An icon in this column indicates that a trace exists for lost or known sinks.

Table 36. Findings table (continued)

Column Heading	Description
Severity	• `High` : Poses a risk to the confidentiality, integrity, or availability of data and/or the integrity or availability of processing resources. High-severity conditions should be prioritized for immediate remediation. • `Medium` : Poses a risk to data security and resource integrity, but the condition is less susceptible to attack. Medium-severity conditions should be reviewed and remedied where possible. • `Low` : Poses minimal risk to data security or resource integrity. • `Info` : The finding, itself, is not susceptible to compromise. Rather, it describes the technologies, architectural characteristics, or security mechanisms used in the code.
Classification	Type of finding: **Definitive** or **Suspect** security finding - or **Scan Coverage** finding. **Note:** In some cases, a classification of **None** may be used to denote a classification that is neither a security finding or a scan coverage finding.
Vulnerability Type	Vulnerability category, such as `Validation.Required` or `Injection.SQL`.
API	The vulnerable call, showing both the API and the arguments passed to it.
Source	A source is an input to the program, such as a file, servlet request, console input, or socket. For most input sources, the data returned is unbounded in terms of content and length. When an input is unchecked, it is considered tainted.
Sink	A sink can be any external format to which data can be written out. Sink examples include databases, files, console output, and sockets. Writing data to a sink without checking it may indicate a serious security vulnerability.
Directory	Full path of the scanned files.
File	Name of the code file in which the security finding or scan coverage finding occurs. File paths in findings are relative to the scanned project working directory.
Calling Method	The function (or method) from which the vulnerable call is made.
Line	Line number in the code file that contains the vulnerable API.
Bundle	Bundle that contains this finding.

Table 36. Findings table (continued)

Column Heading	Description
CWE	ID and topic of the community-developed dictionary of common software weaknesses (Common Weakness Enumeration (CWE) topics).

Customizing the findings table:

In all views with findings, except the Assessment Diff view in AppScan Source for Analysis, you can customize the findings table by identifying only the columns and the column order that you wish to see. Each view may have different settings or you can apply the options to all views. To customize the column order, follow the steps in this task topic.

About this task

To learn about the columns in a findings table, see "Findings table" on page 241.

Procedure

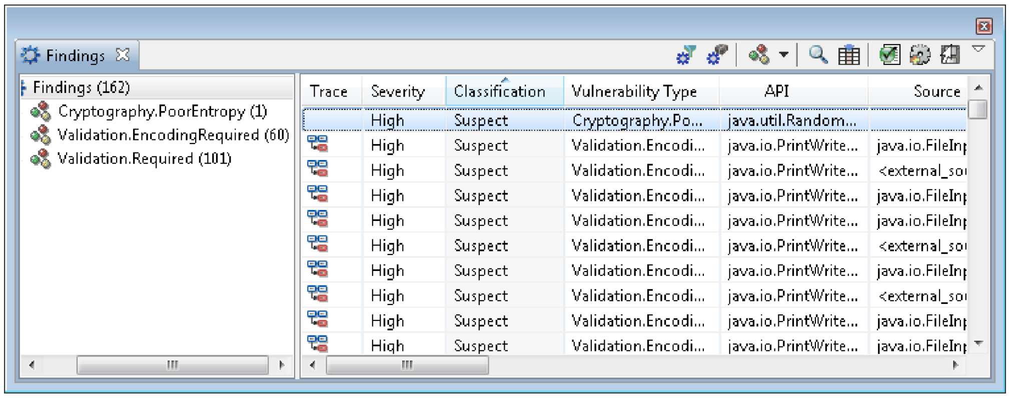
1. Click the **Select and Order Columns** toolbar button.

   Note: In AppScan Source for Development (Visual Studio plug-in), click the **Select and Order Table Columns** toolbar button.
2. In the **Select and Order Columns** dialog box, select the column name, and then click the **Up** arrow or **Down** arrow to move the column location.
3. Click the **Add Column** button to add a column to the view. Alternatively, click the **Delete Column** button to remove a column from the view.

   Note: In AppScan Source for Development (Visual Studio plug-in), these buttons are labelled **Insert** and **Remove** respectively.
4. Click **Restore Defaults** to reset the default columns and column order.
5. Click **OK** to save the settings.

Findings view

The Findings view contains data for findings in an assessment. The findings can be grouped by parameters listed in this topic.



Findings (162)
- Cryptography.PoorEntropy (1)
- Validation.EncodingRequired (60)
- Validation.Required (101)

Trace	Severity	Classification	Vulnerability Type	API	Source
	High	Suspect	Cryptography.Po...	java.util.Random...	
	High	Suspect	Validation.Encodi...	java.io.PrintWrite...	java.io.FileIn
	High	Suspect	Validation.Encodi...	java.io.PrintWrite...	<external_so
	High	Suspect	Validation.Encodi...	java.io.PrintWrite...	java.io.FileIn
	High	Suspect	Validation.Encodi...	java.io.PrintWrite...	java.io.FileIn
	High	Suspect	Validation.Encodi...	java.io.PrintWrite...	<external_so
	High	Suspect	Validation.Encodi...	java.io.PrintWrite...	java.io.FileIn
	High	Suspect	Validation.Encodi...	java.io.PrintWrite...	<external_so
	High	Suspect	Validation.Encodi...	java.io.PrintWrite...	java.io.FileIn
	High	Suspect	Validation.Encodi...	java.io.PrintWrite...	java.io.FileIn

Remember: In AppScan Source for Development (Eclipse plug-in) and AppScan Source for Analysis, these are referred to as *views* in the user interface. In AppScan Source for Development (Visual Studio plug-in), they are called *windows* in the user interface. In this documentation, the term *view* is generally used to denote both *views* and *windows*.

Findings table parameter groupings

In the Findings view, choose the **Select a tree hierarchy** toolbar button down arrow and then choose the parameter by which to group the findings.

Table 37. Findings table parameter groupings

Mode	Grouping
Vulnerability Type	Type, Severity, Classification
Classification	Classification, Severity, Type
File	Project, Directory, File, Method
API	API, Type
Bundle	Bundle, Type, API
CWE	CWE
Table	No grouping

Toolbar buttons

Table 38. Toolbar buttons

Action	Icon	Description
Show findings which do not match the filter	⚙	This button allows you to toggle the display of filtered findings in the Findings view.
Show bundled findings	⚙	This button allows you to toggle the display of bundled findings in the Findings view. This action hides findings in all included bundles that you have created. This setting does not affect the display of findings in excluded bundles - these findings are never shown in the Findings view.
Select a tree hierarchy	Varies depending on the grouping that is selected.	See "Findings table parameter groupings."
Search	🔍	This button opens a dialog box that allows you to search for findings. A variety of search options are available in this dialog box. After a search is conducted, results appear in the Search Results view.

Table 38. Toolbar buttons (continued)

Action	Icon	Description
Select and Order Columns	▦	This button opens the Select and Order Columns dialog box, which allows you to add or remove columns - or modify existing columns.
Report View	✓	This button opens the Report view, which displays the findings according to comprehensive audit reports that measure compliance with software security best practices and regulatory requirements.
Create Custom Finding	⚙	This button is only available in AppScan Source for Analysis. Selecting it opens the Create Custom Finding dialog box, which allows you to add a custom finding to the current assessment.
Save Selected Findings	🖫	If one or more findings are selected, this button opens the Save Selected Findings dialog box, which allows you save the selected findings to a new assessment file.
View Menu		This menu provides quick access to all toolbar button actions.

In the Findings view, you can:
- Open the finding in the code editor
- Create exclusions
- Modify findings
- View findings with different groupings
- Search findings for specific items

When using the view in AppScan Source for Analysis, you can also:
- Move findings to a bundle
- Submit defects to defect tracking systems
- Create custom findings
- Generate findings reports
- Email findings or bundles

Excluded Findings view

The Excluded Findings view contains only excluded findings. An excluded finding is a finding that you omit from scans. In this view, you can search for specific findings. The columns in this view are identical to those in the Findings view.

To re-include findings that have been excluded, follow the instructions in "Re-including findings that have been marked as exclusions" on page 121.

Modified Findings view

The Modified Findings view contains all changed findings of the current application. *Modified findings* are findings with altered vulnerability type, severity, classification, or notes. *Lost findings* (findings not in the currently open assessment) appear in green italics and cannot be modified.

In this view, you can:

- Search for specific findings.
- Make additional modifications

In AppScan Source for Analysis, you can also perform these actions in this view:

- Add findings to bundles
- Submit defects to defect tracking systems
- Email findings (defects)
- Generate findings reports

Fixed/Missing Findings view

The Fixed/Missing Findings view identifies findings that are in bundles but not in the current assessment. A finding is identified as a fixed/missing finding because it was resolved, removed, or the source file was not scanned.

Search Results view

When you search findings, the results appear in the Search Results view.

In this view, you can

- Sort findings
- Edit code in an internal or external editor
- Set the vulnerability type
- Promote suspect and scan coverage findings to definitive
- Set severity level
- Annotate findings
- Exclude specific findings
- Perform subsequent searches

When using the view in AppScan Source for Analysis, you can also:

- Add findings to bundles
- Submit defects to defect tracking systems or email findings
- Generate findings reports

The Search Results view contains only the items that match the search criteria and maintains a maximum of five search results. For example, if you search in the Search Results view for a Vulnerability Type of Buffer Overflow and then for a Classification of Definitive, the search result is the intersection of both searches.

The search criteria appears in the Search field, as a representation of the search as `"<keyword>" in <originating_view>: <fields searched>`, such as `"shutdown" in Findings [Context, API, Method]`. If you close the current assessment, all search results are discarded and the Search field displays contains the text, `No Current Search`.

Report view

The Report view allows you to organize the results of a scan according to a variety of audit reports that measure compliance with software security best practices and regulatory requirements.

The view displays findings according to these reports:

- "CWE/SANS Top 25 2011 report" on page 173
- "DISA Application Security and Development STIG V3R6 report" on page 173
- "Open Web Application Security Project (OWASP) Mobile Top 10 report" on page 174
- "Open Web Application Security Project (OWASP) Top 10 2013 report" on page 173
- "Payment Card Industry Data Security Standard (PCI DSS) Version 1.1 and Version 2.0 reports" on page 174
- "Software Security Profile report" on page 174

If you use AppScan Source for Analysis to create a custom report that is saved to `<data_dir>\reports\profile` (where `<data_dir>` is the location of your AppScan Source program data, as described in "Installation and user data file locations" on page 258), you can also use the Report view to display findings by the custom report.

The columns in the Report view are identical to those in the "Findings view" on page 243.

Searching for findings

In multiple views that contain findings, you can search specific findings. The search criterion includes bundles, code, files, projects, or vulnerability types. The search results appear in the Search Results view.

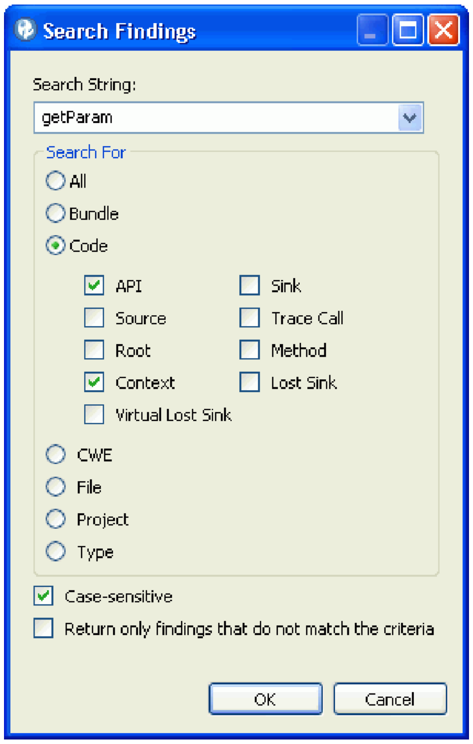
When searching code, the search can be for multiple items or all items, including:

- API
- Context
- Method
- Source
- Sink
- Lost sink
- Root
- trace call

Searching for every occurrence of an item in all findings:

Procedure

1. Select the view in which to search.
2. Select **Edit** > **Find** from the main menu (in AppScan Source for Development (Eclipse plug-in), select **Edit** > **Find/Replace** or in AppScan Source for Development (Visual Studio plug-in), click the **Search** button on a view with findings).


Search Findings

Search String:

getParam

Search For

( ) All
( ) Bundle
(●) Code

[✓] API
[ ] Sink
[ ] Source
[ ] Trace Call
[ ] Root
[ ] Method
[✓] Context
[ ] Lost Sink
[ ] Virtual Lost Sink

( ) CWE
( ) File
( ) Project
( ) Type

[✓] Case-sensitive
[ ] Return only findings that do not match the criteria

3. Type a search string in the **Search Findings** dialog box.
4. Search for the string in a **Bundle**, **Code**, **CWE**, **File**, **Project**, **Type**, or **All**. The matching findings appear in the Search Results view.

   Select **Case-sensitive** to search for case-sensitive text.

   If you are using AppScan Source for Analysis or AppScan Source for Development (Eclipse Plug-in), select **Return only findings that do not match the criteria** to return those findings that do not correspond to the search criteria.

Searching for findings in a findings table:

Procedure

1. Click **Search** on the toolbar.
2. Identify the characteristics for the search and click **OK**.

Searching in a findings tree:

Procedure

1. Click **Search** on the toolbar.
2. Identify the characteristics for the search and click **OK**.

Results

In a findings view, you can also search within a subset of the visible findings. For example, you may want to search for findings in a particular subset, such as Vulnerability Type.

Sources and Sinks view

The Sources and Sinks view provides the ability to view findings based on a trace of input and output.

The Sources and Sinks view is divided into three sections:

- **Sources and Sinks**: In the left panel, there are three top level nodes:
  - **Source**: A source is an input to the program, such as a file, servlet request, console input, or socket. For most input sources, the data returned is unbounded in terms of content and length. When an input is unchecked, it is considered tainted. Sources are listed in any findings table in the **Source** column.
  - **Sink**: A sink can be any external format to which data can be written out. Sink examples include databases, files, console output, and sockets. Writing data to a sink without checking it may indicate a serious security vulnerability.
  - **Lost Sink** A lost sink is an API method that can no longer be traced.

  Each node can be expanded to display affected packages. Packages, in turn, can be expanded to display affected classes and then methods. These methods can then be expanded to display the package, class, and method at the opposite end of the trace. For example, if you are concerned about a particular sink, you can drill down to the method under the **Sinks** root. Once there, the tree underneath that method would show the paths back to all of the sources that led to that sink:

  ```
  - Sources
   - packageA
    - classA
     - methodA
      - packageB
       - classB
        - methodB (at opposite end of trace)
  - Sinks
   - packageB
    - classB
     - methodB
      - packageA
       - classA
        - methodA
  - Lost Sinks
  ```

  The selection that is made in this tree view determines what is displayed in the other two sections of the view.

- **Intermediate nodes**: This section of the view displays the union of all of the intermediate nodes of the traces that apply to the selection in the Sources and Sinks section. It allows you to refine what is displayed in the findings table.

  This section is hidden by default. It can be displayed (or hidden again) by clicking **Show/Hide the intermediate calls table**.

  To only display the findings for a package, class, or method, select the check box in its **Require** column. To filter out the findings for a package, class, or method, select the check box in its **Remove** column. Filter settings made in this section can be used to create a new filter.

  Usage example: Given this tree node in the Sources and Sinks section:

  ```
  - Sources
   - java.util
    - Properties
     - getProperty
  ```

When `getProperty` is selected, the findings table displays only those findings that contain traces that have `getProperty` as the source. At this point, the intermediate nodes section will display all of the intermediate nodes (all nodes in the trace other than the source and sink) for all of the traces that have a source of `getProperty`. However, you may not care if the trace passes through a particular API. For example, you may have a validation routine that ensures that the data coming from `getProperty` is valid, and so you do not want to see traces that go through this validation routine. The intermediate nodes section will include this validation routine, as it is an intermediate node on the trace. You can browse to the validation routine in the intermediate node section and click its **Remove** check box. This will remove all of the findings from the findings table that have traces that pass through this intermediate node.

- **Findings**: This section contains the same "Findings table" on page 241 (and associated actions) that is in the "Findings view" on page 243 and other views with findings. It displays the findings for the sources, sinks, and intermediate nodes that you have chosen to display in the other two sections of the view.

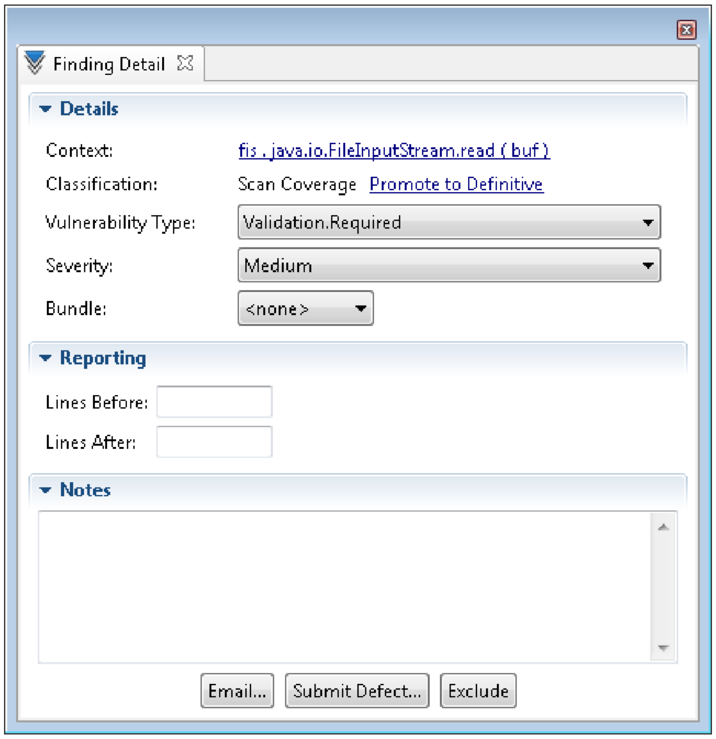
Views that allow you to investigate a single finding

The views in this section are used for investigating single findings.

- "Finding Detail view" on page 127
- "Remediation Assistance view" on page 252
- "Trace view" on page 253

Finding Detail view

When you select a finding, the Finding Detail view displays and allows you to modify its properties. With this view, you can modify an individual finding.



Finding Detail

Details

Context: fis . java.io.FileInputStream.read (buf)

Classification: Scan Coverage Promote to Definitive

Vulnerability Type: Validation.Required

Severity: Medium

Bundle: <none>

Reporting

Lines Before:

Lines After:

Notes

- "Details section" on page 128

- "Reporting section (available in AppScan Source for Analysis and AppScan Source for Development (Eclipse plug-in) only)" on page 128
- "Notes Section" on page 128
- "Finding Detail view actions" on page 129
- "Finding Detail view for custom findings (available in AppScan Source for Analysis only)" on page 129

Details section

- **Context**: Snippet of code that surrounds the vulnerability
- **Classification**: Definitive or Suspect security findings - or Scan Coverage findings - with a link to promote the finding to **Definitive** or revert to the original value if the classification was changed
- **Vulnerability Type**
- **Severity**: High, medium, low, or info
- **Bundle**: Name of the bundle that contains the findings (not available in AppScan Source for Development (Visual Studio plug-in))

Reporting section (available in AppScan Source for Analysis and AppScan Source for Development (Eclipse plug-in) only)

Specify the number of lines of code to include before and/or after the finding in reports.

Notes Section

Annotate the finding.

Finding Detail view actions

- **Exclude**: Click **Exclude** to exclude (remove) the finding from the findings table. To view excluded findings, open the Excluded Findings view.
- Available in AppScan Source for Analysis only:
  - **Email**: If you have configured email preferences, you can email a finding bundle directly to developers to advise them of potential defects found after a scan. The email includes a bundle attachment that contains the findings, and the email text describes the findings.
    1. To email the current finding in the Finding Detail view, click **Email**.
    2. In the Attachment File Name dialog box, specify a name for the finding bundle that will be attached to the email. For example, specifying `my_finding` in the **Attachment File Name** field causes a bundle with file name `my_finding.ozbdl` to be attached to the email.
    3. Click **OK** to open the Email Findings dialog box. By default, the **Mail To** field in the Email Findings dialog box will populate with the **To Address** that is specified in the email preferences - however, it can easily be changed when preparing the email. In this dialog box, review the contents of the email and then click **OK** to send the email.
  - **Submit Defect**: To submit the finding as a defect, click **Submit Defect**. This opens the Select Defect Tracking System dialog box.
    - If you select **ClearQuest** and click **OK**, the Attachment File Name dialog box opens. In it, specify a name for the finding bundle that will be attached to the defect and then click **OK**. Log in to Rational ClearQuest and submit the findings.

- If you select **Quality Center** and click **OK**, the Login dialog box opens, allowing you to log in to Quality Center to submit the findings.
- If you select either **Team Foundation Server** option, a dialog box opens, prompting you to log into the defect tracking system and provide other configuration details.

Note: Rational Team Concert is the only supported defect tracking system on OS X.

Finding Detail view for custom findings (available in AppScan Source for Analysis only)

The Finding Detail view for custom findings provides additional information that you can edit:

- File
- Line
- Column
- API

In addition, the method by which you edit the "Details section" on page 128 is different than standard findings for some fields (for example, the classifications for custom findings appear in a list).

Remediation Assistance view

The AppScan Source Security Knowledgebase provides context-specific intelligence for each vulnerability. The Knowledgebase tells you what the vulnerability is, why it is insecure, how to fix it, and how to avoid it in the future. Once you scan source code, the Knowledgebase provides the specific information needed to eliminate the risk from your mission-critical applications. Knowledgebase remediation advice appears in the Remediation Assistance view. Once you scan, the Knowledgebase provides the specific information needed to eliminate the risk from your mission-critical applications.

To view the Knowledgebase and obtain remediation advice

- Select a finding in a findings table, and then open the Knowledgebase Help or Remediation Assistance view.
- In AppScan Source for Analysis, you can also select **Help** > **Security Knowledgebase** from the menu to see the entire Knowledgebase.

Specific APIs in the database list the severity level and the severity type. For example, the API, `strcpy()`, a Buffer Overflow type, has a High severity level. The description states that `strcpy()` is susceptible to destination buffer overflow because it does not know the length of the destination buffer and therefore cannot check to make sure it does not overwrite it. Fix this problem by using `strncpy ()`, which takes a length parameter.

If the finding has an associated Common Weakness Enumeration (CWE) ID, from the Remediation Assistance view, a hyperlink to the CWE topic (`CWE: <id>`) at `http://cwe.mitre.org/data/definitions/<CWE_ID>.html` appears.

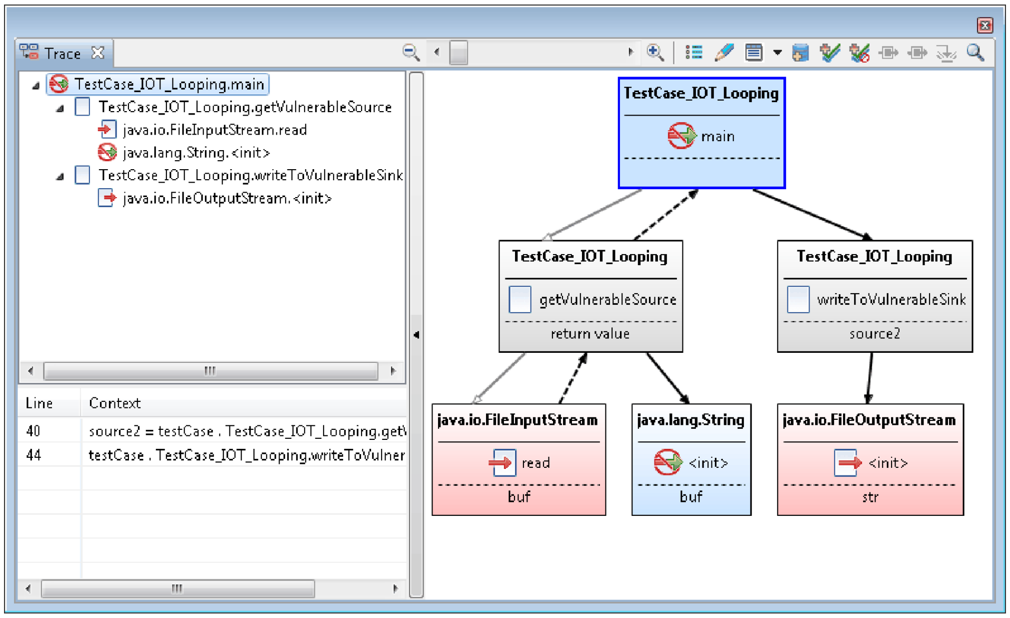
Trace view

AppScan Source performs input/output analysis and identifies and displays these vulnerabilities. An icon appears in the findings list to identify the row that contains an AppScan Source trace graph.

In the Trace view, you see the root node, where the input and output stacks meet. The input stack is a series of calls that lead to a *source* known to provide tainted data. The output stack is a series of calls that lead to a *sink*. An AppScan Source trace is generated when the code analyzed can track the use of an unprotected source to an unprotected sink.

- **Source**: A source is an input to the program, such as a file, servlet request, console input, or socket. For most input sources, the data returned is unbounded in terms of content and length. When an input is unchecked, it is considered tainted. Sources are listed in any findings table in the **Source** column.
- **Sink**: A sink can be any external format to which data can be written out. Sink examples include databases, files, console output, and sockets. Writing data to a sink without checking it may indicate a serious security vulnerability.
- **Lost Sink** A lost sink is an API method that can no longer be traced.

This diagram illustrates the call sequence from the root through the input stack and the output stack.



- TestCase_IOT_Looping.main
  - TestCase_IOT_Looping.getVulnerableSource
    - java.io.FileInputStream.read
    - java.lang.String.<init>
  - TestCase_IOT_Looping.writeToVulnerableSink
    - java.io.FileOutputStream.<init>

Line	Context
40	source2 = testCase . TestCase_IOT_Looping.get\
44	testCase . TestCase_IOT_Looping.writeToVulner

In the diagram:

- Unfilled arrows show a call that does not have a known tainted data flow.
- Filled arrows have potentially tainted data. Dashed lines show a return path.
- Solid lines represent a method call.

Tip:

- In the Trace view, hovering over trace nodes in the graph provides information about the node.
- The two left panels in the view (the input/output stacks panel and the data flow panel) can be collapsed for easier viewing of the graphical call graph. To collapse these panels, select the **Hide tree view** arrow button. To display these panels when they are hidden, select the **Show tree view** arrow button.

- Move the scroll bar to zoom in and focus on details - or to zoom out to see more. Hovering over the zoom scroll bar provides the current zoom level. To zoom in to the maximum level, click **Zoom to 200%**. To zoom out as far as possible, click **Zoom to fit**.

Views that allow you to work with assessments

The views in this section are used for working with assessments at a high level.

- "Assessment Summary view"
- "Filter Editor view" on page 255
- "Vulnerability Matrix view" on page 255

Assessment Summary view

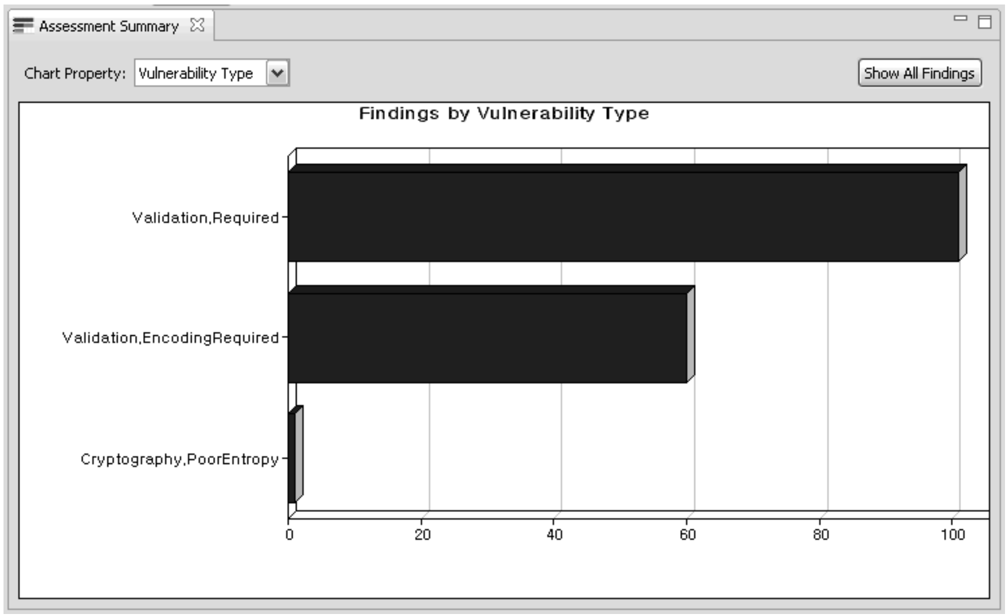
The Assessment Summary view, a bar chart graphical view of the open assessment, displays information for select findings.

Note:

- The Assessment Summary view is not available on OS X.
- In AppScan Source for Development (Visual Studio plug-in), this view is part of the Edit Filters window.

You can view by a chart property:

- **Vulnerability Type**: Vulnerability type, such as `Validation.Encoding` or `Injection.SQL`
- **API**: API name in which the vulnerability occurs
- **Project**: Findings by project if more than one project exists
- **File**: Individual file in which the vulnerabilities appear


Assessment Summary
Chart Property: Vulnerability Type
Show All Findings
Findings by Vulnerability Type
Validation,Required
Validation,EncodingRequired
Cryptography,PoorEntropy
0
20
40
60
80
100

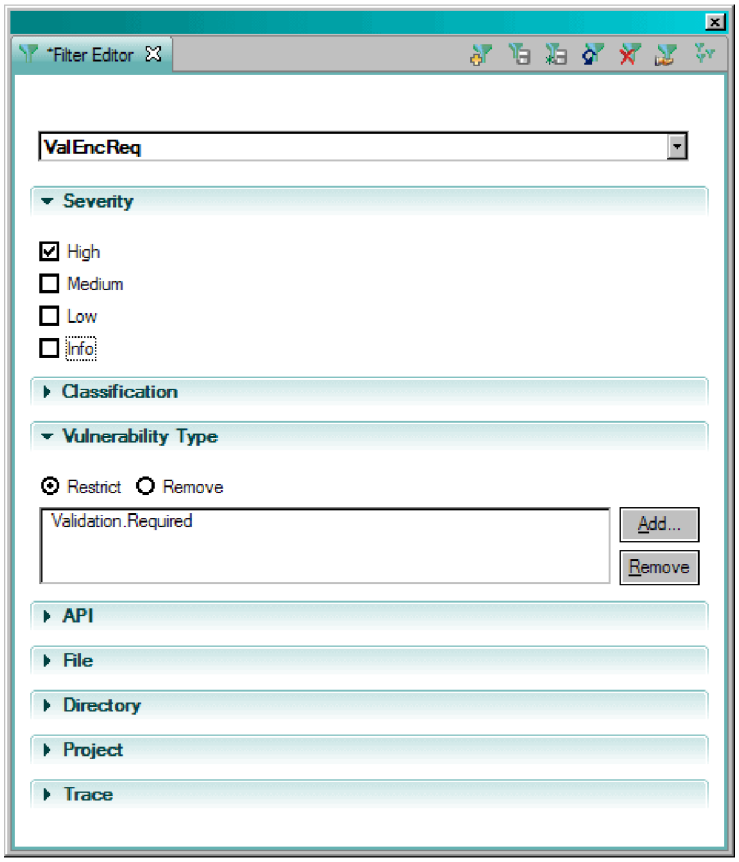
Click the chart to drill down to findings details and begin triage.

Tip: Hovering over a graph bar in the Assessment Summary view provides the exact number of findings represented by the bar.

Filter Editor view

The Filter Editor view provides a more granular manipulation of the currently selected filter than other AppScan Source views. This view consists of all criteria on which you can filter.

Note: In AppScan Source for Development (Visual Studio plug-in), this view is part of the Edit Filters window.

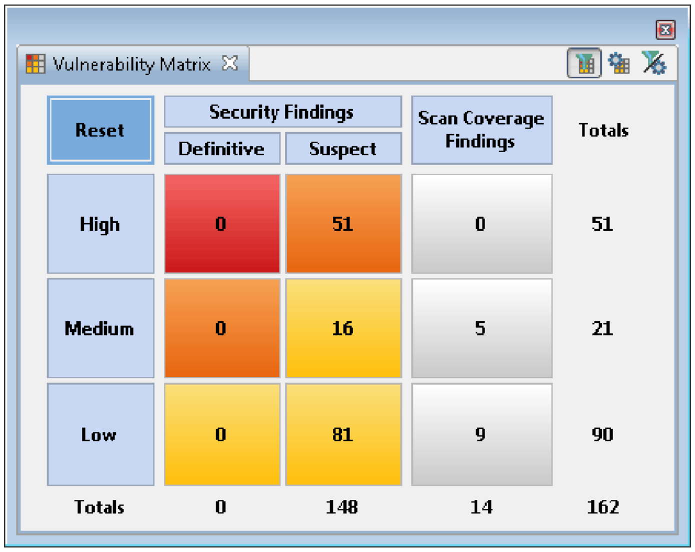


Tip: In the Filter Editor view **Trace** section, hovering over a trace entry provides details about the entry.

Vulnerability Matrix view

The Vulnerability Matrix view displays the aggregate number of findings for all applications included in the scan. Modifications to findings update the matrix.

Note: In AppScan Source for Development (Visual Studio plug-in), this view is part of the Edit Filters window.



Reset	Security Findings		Scan Coverage Findings	Totals
	Definitive	Suspect		
High	0	51	0	51
Medium	0	16	5	21
Low	0	81	9	90
Totals	0	148	14	162

Security findings and scan coverage findings appear in colored squares that indicate the order of priority in which findings should be investigated or dealt with:

1. High severity definitive security findings are colored red, marking them as the highest priority.
2. Medium severity definitive and high severity suspect security findings are colored orange and should be dealt with next.
3. These matrix entries are colored yellow, and should be considered next:
   - Low severity definitive security findings
   - Medium and low severity suspect security findings
4. Scan coverage findings are in grey squares and can be given the lowest priority.

When you click a cell, row header, or column header in the **Vulnerability Matrix**, it updates the current filter to include only the results in that cell, row, or column. Click **Reset** to return to a view of all findings.

In the Vulnerability Matrix view, toolbar buttons control the numbers in the colored squares. You can view:

- Counts and totals of filtered findings only
- Counts and total number of findings

  Note:

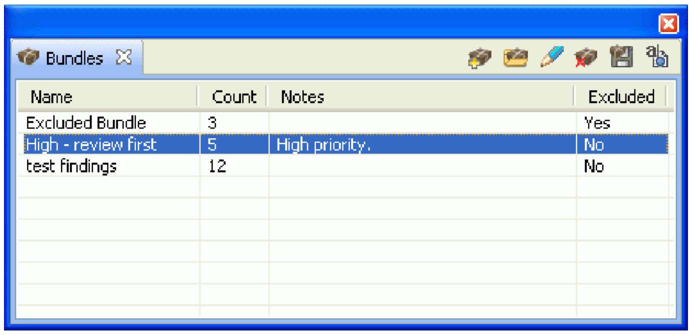
  – Findings that are classified with the **Info** severity level are not included in the Vulnerability Matrix view.

  – If you open a scan that was run in a product that supports quality analysis, findings related to quality are not reflected in the Vulnerability Matrix view.
- Counts and total number of findings that are filtered and all findings

Note: Filters that are applied outside of the Vulnerability Matrix view may not affect the Vulnerability Matrix view. The Vulnerability Matrix view **Show the counts of filtered findings** toolbar button must be selected for the filter to be reflected in the Vulnerability Matrix view.

Bundles view

In the Bundles view, you create new bundles, add findings to a bundle, view bundles and notes, rename, or delete a bundle. This view lists the bundle name, any notes attached to the bundle, the number of findings in the bundle, and if the bundle is excluded. Once you open the bundle to see its contents, you can move findings to other bundles, modify the findings, edit the code, or submit the bundle to a defect tracking system.



Name	Count	Notes	Excluded
Excluded Bundle	3		Yes
High - review first	5	High priority.	No
test findings	12		No

For more information, see “Triage with bundles” on page 122.

Bundle view

The Bundle View displays the findings in a bundle. Bundles are sets of findings created in AppScan Source for Analysis.

To view the findings in a bundle, double-click a bundle name in the Bundles View. The bundle name appears as the title in the Bundle View. You can also import a bundle and view its contents in the Bundle View. You cannot modify or delete findings in a bundle.

The Bundle View, similar to a findings table contains the following detailed information:

Table 39. Bundle view columns

Column	Description
Trace	An icon in this column indicates that a trace exists for lost or known sinks.
File	Name of the code file in which the security finding or scan coverage finding occurs. File paths in findings are relative to the scanned project working directory.
Classification	Type of finding: **Definitive** or **Suspect** security finding - or **Scan Coverage** finding. **Note:** In some cases, a classification of **None** may be used to denote a classification that is neither a security finding or a scan coverage finding.

Table 39. Bundle view columns (continued)

Column	Description
Severity	• High : Poses a risk to the confidentiality, integrity, or availability of data and/or the integrity or availability of processing resources. High-severity conditions should be prioritized for immediate remediation. • Medium : Poses a risk to data security and resource integrity, but the condition is less susceptible to attack. Medium-severity conditions should be reviewed and remedied where possible. • Low : Poses minimal risk to data security or resource integrity. • Info : The finding, itself, is not susceptible to compromise. Rather, it describes the technologies, architectural characteristics, or security mechanisms used in the code.
Vulnerability Type	Vulnerability category, such as `Validation.Required` or `Injection.SQL`.
Context	Snippet of code that surrounds the vulnerability.
Calling Method	The function (or method) from which the vulnerable call is made.
CWE	ID and topic of the community-developed dictionary of common software weaknesses (Common Weakness Enumeration (CWE) topics).
Line	Line number in the code file that contains the vulnerable API.
Notes	Any notes added to this finding.
Defect ID	Defect ID from a defect tracking system.

Installation and user data file locations

When you install AppScan Source, user data and configuration files are stored outside of the installation directory.

- "Default installation location"
- "Default AppScan Source data directory" on page 259
- "AppScan Source temporary file location" on page 259

Default installation location

When AppScan Source is installed, the software is placed in one of these default locations:

- 32-bit versions of Microsoft Windows:
  `<SYSTEMDRIVE>:\Program Files\IBM\AppScanSource`
- 64-bit versions of Microsoft Windows:
  `<SYSTEMDRIVE>:\Program Files (x86)\IBM\AppScanSource`

- Linux: If you are the root user, the Installation Wizard installs your software in `/opt/ibm/appscansource`. If you are not the root user, you can install the AppScan Source for Development Eclipse plug-in - which installs to `<home_directory>/AppScan_Source` by default.
- OS X: `/Applications/AppScanSource.app`

Important:

- The installation directory name can only contain English characters. Folders with names containing non-English characters are not permitted.
- If you are installing on Windows, you must have Administrator privileges to install AppScan Source components.
- If you are installing on Linux, you must have root privileges to install AppScan Source server components.

Default AppScan Source data directory

AppScan Source data consists of items such as configuration, sample, and log files. When AppScan Source is installed, data files are placed in these locations by default:

- Microsoft Windows: `<SYSTEMDRIVE>:\ProgramData\IBM\AppScanSource`

  Note: `ProgramData\` is a hidden folder, and to see it you must modify your view preferences in **Explorer** to show hidden files and folders.
- Linux: `/var/opt/ibm/appscansource`
- OS X: `/Users/Shared/AppScanSource`

To learn how to change the location of the AppScan Source data directory, see "Changing the AppScan Source data directory."

AppScan Source temporary file location

Some AppScan Source operations result in the creation of temporary files, which are stored in these locations by default:

- Microsoft Windows: `<SYSTEMDRIVE>:\ProgramData\IBM\AppScanSource\temp`

  Note: `ProgramData\` is a hidden folder, and to see it you must modify your view preferences in **Explorer** to show hidden files and folders.
- Linux: `/var/opt/ibm/appscansource/temp`
- OS X: `/Users/Shared/AppScanSource/temp`

The temporary file location is always located in a `temp` directory in the AppScan Source data directory. You can change the temporary file location by changing the data directory, as described in "Changing the AppScan Source data directory." This will cause the `temp` to be located in the data directory that you have chosen.

Changing the AppScan Source data directory

You may want to change the location of the AppScan Source data directory for the purpose of managing hard disk space. You can change the location after AppScan Source installation by following the steps in this topic.

Before you begin

Before completing this task, ensure that all AppScan Source client applications have been exited or shut down. AppScan Source client applications include:

- AppScan Source for Analysis
- AppScan Source for Development (Eclipse or Visual Studio plug-in)(supported only on Windows and Linux)
- AppScan Source command line interface (CLI)
- AppScan Source for Automation

In addition, if you have installed AppScan Source for Automation, ensure that the Automation Server has been shut down:

- On Windows, stop the **IBM Security AppScan Source Automation** service.
- On Linux, issue this command: `/etc/init.d/ounceautod stop`
- On OS X, issue this command: `launchctl stop com.ibm.appscan.autod`

Procedure

1. Define an `APPSCAN_SOURCE_SHARED_DATA=<data_dir>` environment variable, where `<data_dir>` is the location in which you want AppScan Source data to be stored.

   Note:

   - The `<data_dir>` location must be a complete and absolute path that already exists on the same machine as your AppScan Source installation.
   - The `<data_dir>` directory name can only contain English characters. Folders with names containing non-English characters are not permitted.
2. Locate the default data directory that was created when AppScan Source was installed (see "Default AppScan Source data directory" on page 259 to learn about default data directory locations).
3. Copy or move the contents of the default data directory to the `<data_dir>` location that is specified in the environment variable.
4. **Applies only to AppScan Source for Automation installed on Linux**:
   a. Edit the `/etc/init.d/ounceautod` file.
   b. Locate this line,

      ```
      su - ounce -c
      'export LD_LIBRARY_PATH="/opt/IBM/AppScan_Source/bin":$LD_LIBRARY_PATH &&
      cd "/opt/IBM/AppScan_Source/bin" &&
      "/opt/IBM/AppScan_Source/bin/ounceautod" -s' >>
      "/var/opt/ibm/appscansource/logs/ounceautod_output.log" 2>&1 &
      ```

      and replace it with this:

      ```
      su - ounce -c
      'export APPSCAN_SOURCE_SHARED_DATA=<new data directory path here> &&
      export LD_LIBRARY_PATH="/opt/IBM/AppScan_Source/bin":$LD_LIBRARY_PATH &&
      cd "/opt/IBM/AppScan_Source/bin" &&
      "/opt/IBM/AppScan_Source/bin/ounceautod" -s' >>
      "<new data directory path here>/logs/ounceautod_output.log" 2>&1 &
      ```

      Note: The above command is one line.
   c. Save the `/etc/init.d/ounceautod` file.

What to do next

If you have installed AppScan Source for Automation, start the Automation Server:

- On Windows, start the **IBM Security AppScan Source Automation** service.
- On Linux, issue this command: `/etc/init.d/unceautod start`
- On OS X, issue this command: `launchctl start com.ibm.appscan.autod`

Glossary

This glossary includes terms and definitions for AppScan Source.

The following cross-references are used in this glossary:
- See refers you from a term to a preferred synonym, or from an acronym or abbreviation to the defined full form.
- See also refers you to a related or contrasting term.

To view glossaries for other IBM products, go to www.ibm.com/software/globalization/terminology.

A

application
One or more computer programs or software components that provide a function in direct support of a specific business process or processes.

assembly
A collection of types and resources that form a unit of deployment, version control, reuse, activation scoping, and security permissions in .NET Framework applications.

assessment
A collection of findings as a result of scanned code that a user can work with, save, and share with other people.

attack Any attempt by an unauthorized person to compromise the operation of a software program or networked system.

attribute
A characteristic of an application that helps organize the scan results into meaningful groupings, such as by department or project leader.

B

bundle
A set of findings that the user creates. Bundles can be exported and shared between people and applications.

C

callback
A way for one thread to notify another application thread that an event has happened.

call graph
A graph that uses lines represents the flow of data between subroutines in a program.

cross-site scripting
An attack technique that forces a website to echo client-supplied data, which execute in a user's Web browser.

D

defect A type of change request that identifies an anomaly or flaw in a work product.

E

encode
In computer security, to convert plaintext into an unintelligible form by means of a code system.

exception
An indication of a suspicious and potentially vulnerable condition that requires additional information or investigation.

exclusion
A finding that a user can mark and ignore.

F

filter A set of rules that defines findings with certain traits.

finding
The discovery of an instance of a security exposure in code. AppScan divides findings into two categories: vulnerability and exception.

L

lost sink
An API method that can no longer be traced.

P

perspective
A group of views that show various aspects of the resources in the workbench.

R

remediation
A suggestion for how to fix an issue.

S

scan The process of AppScan exploring and testing an application and providing the results.

scan rule
A pattern or regular expression that is searched during a scan.

sink Any external format to which data can be written out. Sink examples include database, files, console output, and sockets.

socket A communications handle used by TCP/IP.

stack An area in memory that typically stores information such as temporary register information, values of parameters, and return addresses of subroutines and is based on the principle of last in, first out (LIFO).

T

taint	Insecure data that is allowed to flow through the code.
triage	The process of evaluating findings and determining how to resolve them.

V

V-Density
A numerical expression that enables a consistent way to evaluate the vulnerability of your applications. V-Density is calculated by relating the number and criticality of vulnerabilities and exceptions to the size of the application or project being analyzed.

vulnerability analysis cache
A cache of vulnerabilities found during a scan of source code that can be used for subsequent scans to reduce scan time.

W

workbench
The user interface and integrated development environment (IDE) in Eclipse and Eclipse-based tools such as IBM Rational Application Developer.

X

XSS	See cross-site scripting.

Legal notices

Index

Printed in USA