



**Hewlett Packard
Enterprise**

HPE Fortify Software Security Assurance

Jeffrey Hsiao
Security Solutions Architect
Jeffrey.Hsiao@hpe.com

Haleh Nematollahy
Sr. Security Solutions Architect
Haleh.Nematollahy@hpe.com



Agenda

- Introductions
- Application Security Challenges
- HPE Fortify Solution
- HPE Fortify SCA Overview and Exercises
- HPE Fortify SSC Overview and Exercises
- Lunch
- HPE WebInspect Overview and Exercises
- HPE WebInspect Enterprise Overview
- Wrap-Up

Introductions

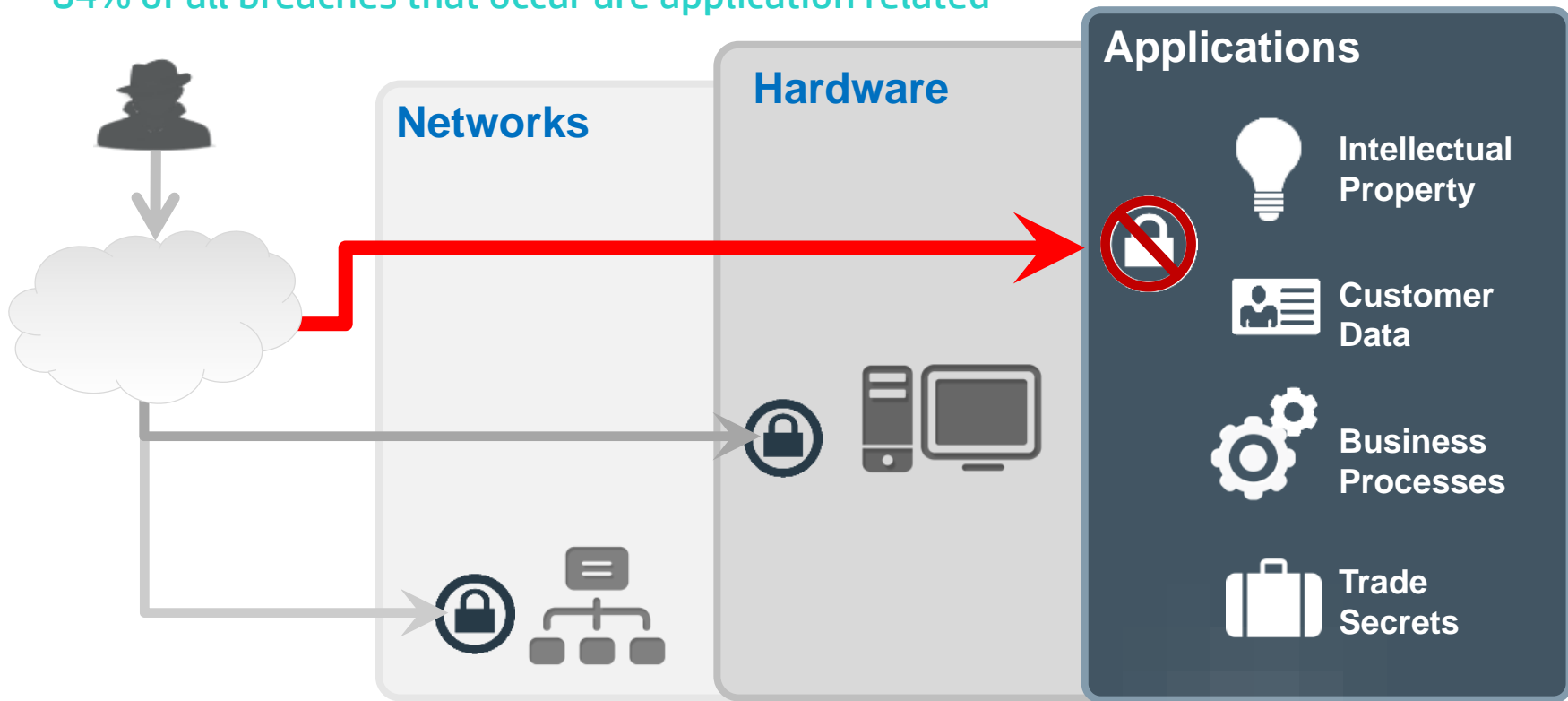
- Name and organization
- Role and duties
- Secure coding background

The Software Security Problem



Cyber attackers are targeting applications

84% of all breaches that occur are application related



Vulnerabilities in Software

What is a software or application vulnerability?

A vulnerability is a hole or a weakness in the application, which can be a design flaw or an implementation bug, that allows an attacker to cause harm to the stakeholders of an application.



So How Bad Can It Be?

Vulnerabilities in Software

OWASP

–The **O**pen **W**eb **A**pplication **S**ecurity **P**roject is a worldwide free and open community focused on improving the security of application software.

–www.owasp.org

–This community routinely publishes a list of the top-10 application security vulnerabilities.

–New list published in 2013.

–Previous list was published in 2010.



OWASP Top 10

As of 2013, OWASP lists the following top-10 categories:

- 1) Injection
- 2) Broken Authentication and Session Management
- 3) Cross-Site Scripting (XSS)
- 4) Insecure Direct Object Reference
- 5) Security Misconfiguration
- 6) Sensitive Data Exposure
- 7) Missing Function Level Access Control
- 8) Cross Site Request Forgery (CSRF)
- 9) Using Components with Known Vulnerabilities
- 10) Unvalidated Redirects and Forwards



OWASP A1 – Injection Flaws

- Multiple Types of Vulnerabilities
 - SQL Injection
 - LDAP Injection
 - XPath Injection
 - XSLT Injection
 - HTML Injection
 - OS Command Injection
- Basic Definition: Interpreters execute unintended commands on a server

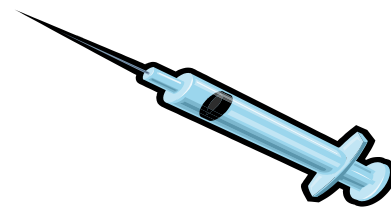
OWASP A1 – Injection Flaws

- SQL Injection – The Worst of the Worst
 - Has been on the top of the OWASP Top 10 since the beginning
 - The favorite vulnerability of Anonymous, LulzSec, and Black Hats
 - Pretty easy to detect and exploit
 - Easy access to your data
- What is it?
 - Data from an untrusted source is mixed into a SQL query allowing an attacker to change the query.

Injection Attack

General Algorithm

1. Generate dynamic string to be used by an interpreter
 - Append raw data to string
 - Raw data is unexpected
2. Pass string to interpreter to be executed
3. Interpreter performs some other operation as a result of unexpected data



SQL Injection Attack

Conceptual Example and Flow

1. Perform a dynamic query against a SQL database such as:

```
Select * from USERS where name = '+userName+'
```

2. User sets `userName = 'x'; drop table members; --'`

3. SQL database query has now changed meaning:

```
Select * from USERS where name = 'x'; drop table members; --'
```

4. Database will now delete the *members* table instead of querying user table

Preventing SQL Injections

Another Exploit - incorrect filtered escape character

1) SQL Statement intended to retrieve the user's account transactions:

```
String query = "SELECT * FROM acct_trans WHERE acct_num = '\" +  
    request.getParameter("AcctNum") + \"'";
```

...

2) Exploit: AcctNum = '12345' or '1' = '1' --

3) Result: SELECT * FROM acct_trans

```
WHERE acct_num = '12345' or '1' = '1' --
```

How Do I Fix SQL Injection?

- Input Validation – Yes
 - Detect unauthorized input before processed by application
- Parameterized Queries (prepared stmt) – Even Better
 - Define SQL code and then pass in each parameter to the query later
 - Parameters are placeholders for values

Parameterized Query

```
String selectStatement = "SELECT * FROM User WHERE userId = ? ";  
PreparedStatement prepStmt = con.prepareStatement(selectStatement);  
prepStmt.setString(1, userId);  
ResultSet rs = prepStmt.executeQuery();
```

NOT

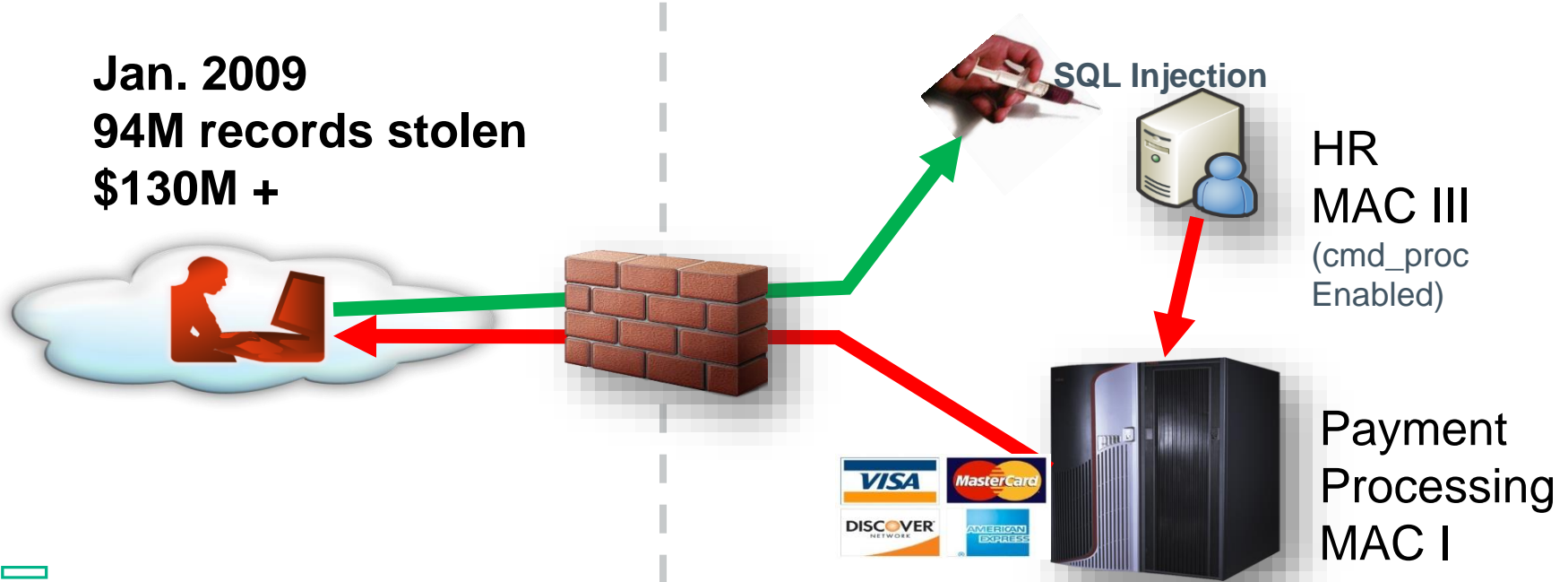
```
String strUserName = request.getParameter("Txt_UserName");  
PreparedStatement prepStmt = con.prepareStatement("SELECT * FROM user WHERE userId = '"+strUserName+'");
```


Injection Attack Costly Example

Heartland Payment Systems – Jan, 2009



Jan. 2009
94M records stolen
\$130M +



One Vulnerability To Rule Them All

Heartland Payment Systems

- The method used to compromise Heartland's network was ultimately determined to be SQL injection. Code written eight years ago for a web form allowed access to Heartland's corporate network. This code had a vulnerability that (1) was not identified through annual internal and external audits of Heartland's systems or through continuous internal system-monitoring procedures, and (2) provided a means to extend the compromise from the corporate network to the separate payment processing network. Although the vulnerability existed for several years, SQL injection didn't occur until late 2007.... the intruders spent almost six months and many hours hiding their activities while attempting to access the processing network, bypassing different anti-virus packages used by Heartland. After accessing the corporate network, the fraudsters installed sniffer software that was able to capture payment card data, including card numbers, card expiration dates, and, in some cases, cardholder names as the data moved within Heartland's processing system.

Heartland Payment Systems: Lessons Learned from a Data Breach

Julia S. Cheney, January 2010

Federal Reserve Bank of Philadelphia

SQL Injection Responsible for Major Losses

As reported earlier, LulzSec and Anonymous use a hacking technique called SQL injection (SQLi) to breach systems. Thursday Imperva pointed to a recent report (PDF) stating that, since July, web applications are attacked by using SQL injection an average of 71 times per hour. Even more, specific applications were occasionally under aggressive attacks and at their peak, were attacked 800 to 1300 times per hour.

"SQL injection is the most pernicious vulnerability in human computer history," Imperva said in a blog earlier this week. "From 2005 through today, SQL injection has been responsible for 83-percent of successful hacking-related data breaches. Using data from Privacyrights.org, we checked the data breaches from 2005 to today. There were 312,437,487 data records lost due to hacking with about 262 million records from various breaches including TJMax, RockYou and Heartland, all of which were SQL injection attacks."

Tom's Hardware citing Imperva blog

<http://www.tomshardware.com/news/LulzSec-Anonymous-SQL-Injection-SQLi-Imperva,13513.html>

Exercise 1: Start the Fortify Demo

Environment Setup

- Start the Fortify Demo Server
- There's a "Launch the Riches Demo App" Shortcut on your desktop

- **It should already be started. You Should see some Command Prompt Windows.

Demo

- Open Internet Explorer and browse to <http://localhost:8080/riches> (there should also be a shortcut)
- Click the Locations Button at the top.
- There is SQL Injection in this form. See if you can find it!
- Valid Zip Codes (94404, 10005, 94123)

OWASP A3 – Cross Site Scripting (XSS)

- The Most Prevalent Vulnerability on the OWASP Top 10
 - Was at the top until the ranking methodology changed
 - Very easy to detect and exploit
- What is it?
 - Your application is used as a platform to attack your users.
 - Allows an attacker to insert malicious code into the user's session
 - Used to deface web sites, hijack sessions, steal credentials, and install malware

OWASP A3 – Cross Site Scripting (XSS)

- Reflected XSS
 - Requires a user to execute an action that contains the attack payload. (Such as clicking a link in a phishing email)
 - The attack only affects the user that executes the action
- Persistent XSS
 - Attack payload is injected into a data store, such as a database.
 - The attack affects every user that uses the application.
 - The most impactful variant of XSS

Preventing XSS – Input/Output Validation

- Blacklisting – Developing a naughty list of characters/tags
 - Nearly impossible to write black lists that cover all attack vectors
 - Many ways to obfuscate attack payloads
 - Using case, null characters, and flaws in browser rendering
 - Using alternate tags, such as IMG, IFRAME, links, body, etc
 - Using alternate encodings and languages
 - Check out:
https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet
- Whitelisting – Using regular expressions
 - [A-Za-z0-9]{5,25} – Possible regex for a username

Preventing XSS – Output Encoding

- Encoding – Making Malicious Code Unexecutable
 - `<script>` becomes `<script>`
 - HTML Encoding for all data rendered in plain HTML
 - Special care should be taken for data inserted into JavaScript and as tag attributes
- Many Standard Encoding Libraries are not sufficient
 - Use the AntiXSS Library from Microsoft for .NET (now included in 4.5)
 - OWASP Enterprise Security API (ESAPI)

Cross Site Scripting Famous Example

PayPal, circa 2004 - 2006

- Steal credit card numbers
 1. Users access URL on genuine PayPal site
 2. Page modified via XSS attack to silently redirect user to external server
 3. Fake PayPal Member log-in page
 4. User supplies login credentials to fake site
- Exploitable for two years



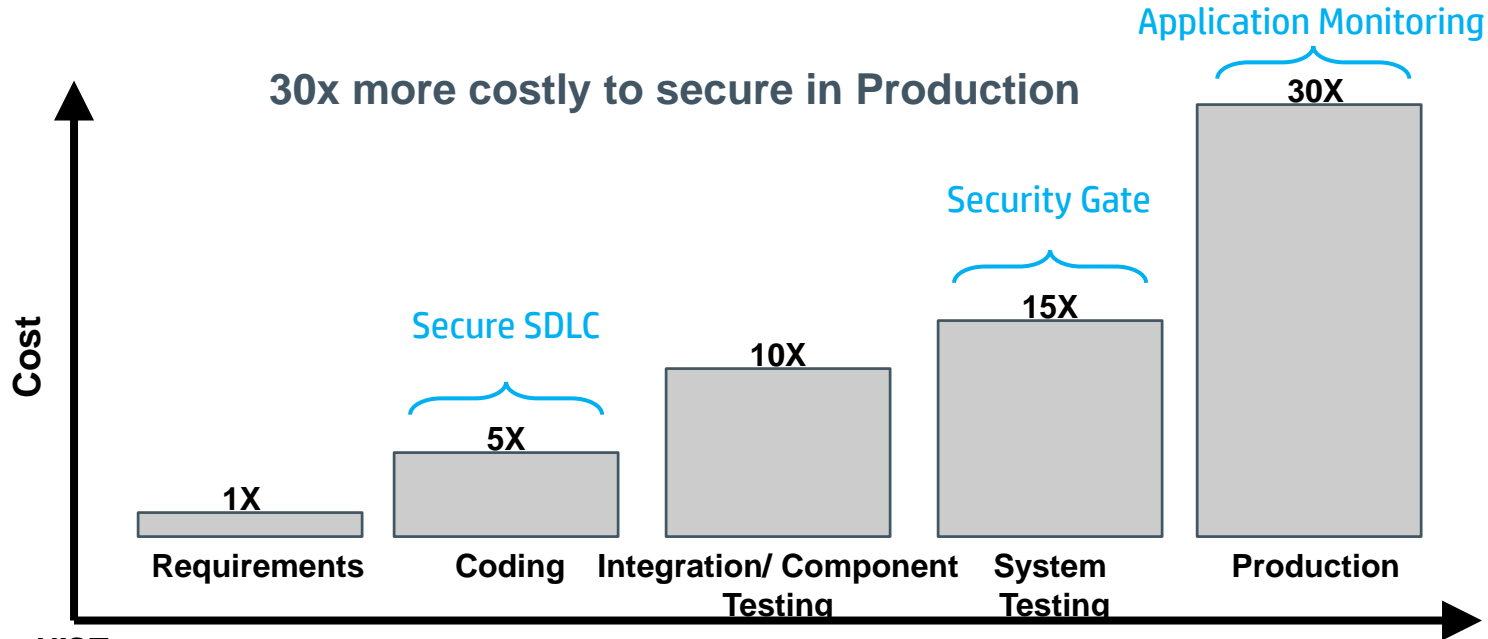
Exercise 2

- Click the submit button on the login form.
- Open Internet Explorer and browse to <http://localhost:8080/riches> (there should also be a shortcut)
- There is Cross Site Scripting in the login page. See it?
- Valid Login (eddie/eddie)

The Solution



Software Security Assurance (SSA & SDLC)

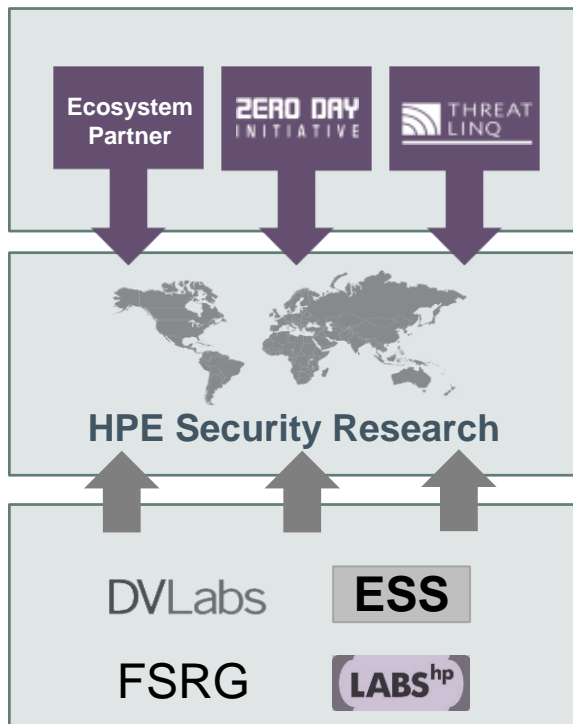




HPE Fortify Solutions

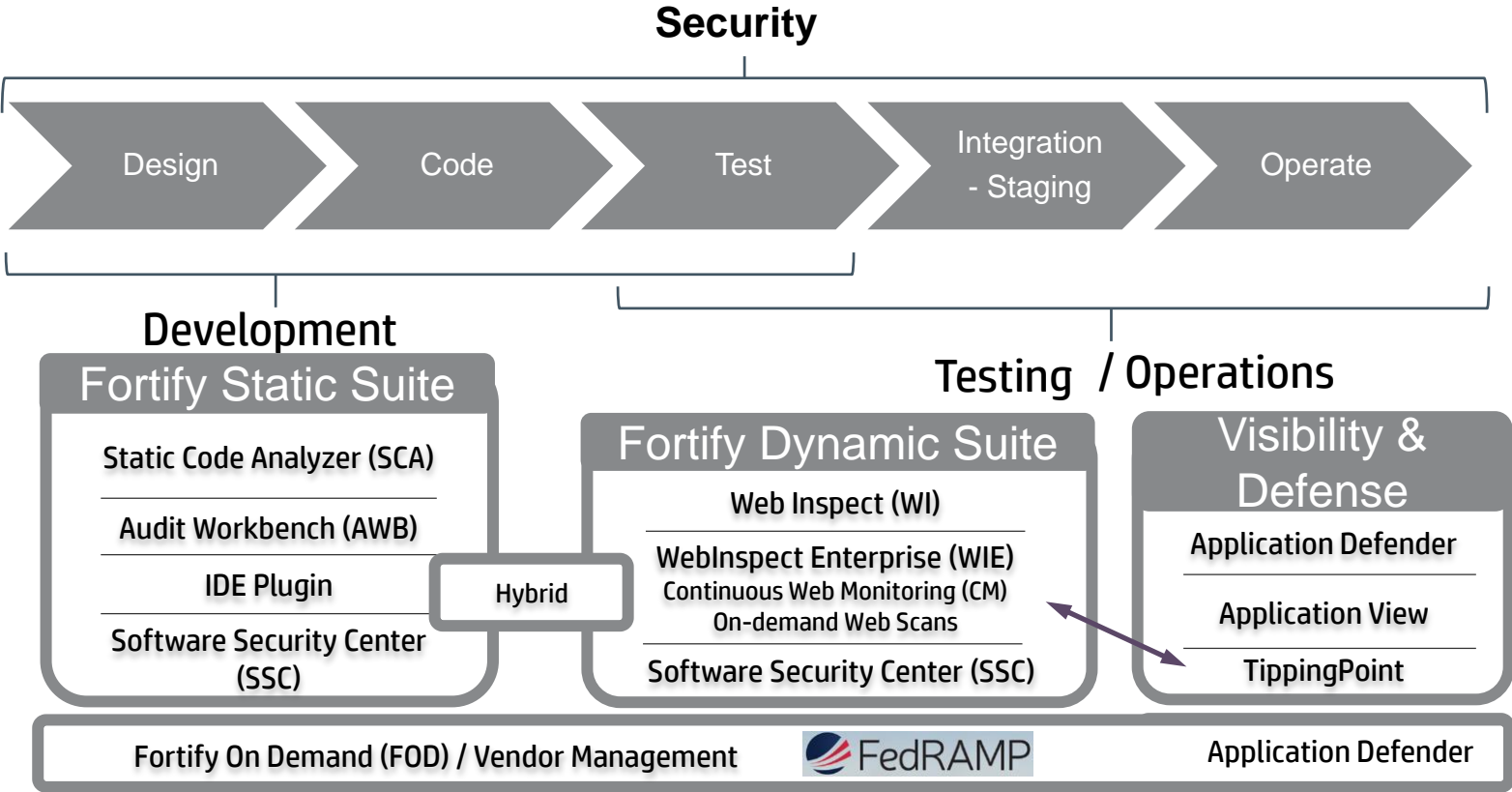
Security solutions backed by HPE Security Research

Actionable Security Intelligence



- SANS, CERT, NIST, OSVDB, software & reputation vendors
 - 3000+ Researchers
 - 2000+ Customers sharing data
-
- Largest commercial IT security research group
 - Continuously finds more vulnerabilities than the rest of the market combined (75% of publicly reported critical vulnerabilities)
 - Frost & Sullivan winner for vulnerability research last three years
-
- Collaborative effort of market leading teams: DV Labs, ArcSight, Fortify, HPE Labs, Application Security Center
 - Collect network and security data from around the globe

Software Security Assurance (SSA & SDLC)

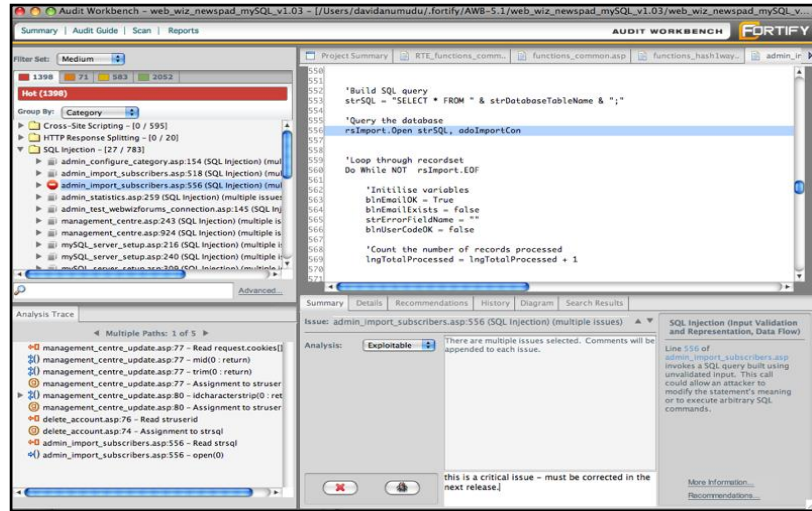




HPE Fortify SCA

HPE Fortify Static Code Analyzer (SCA)

Securing your application code in development



Problem it solves:

Identifies all risks in the source code for applications in development

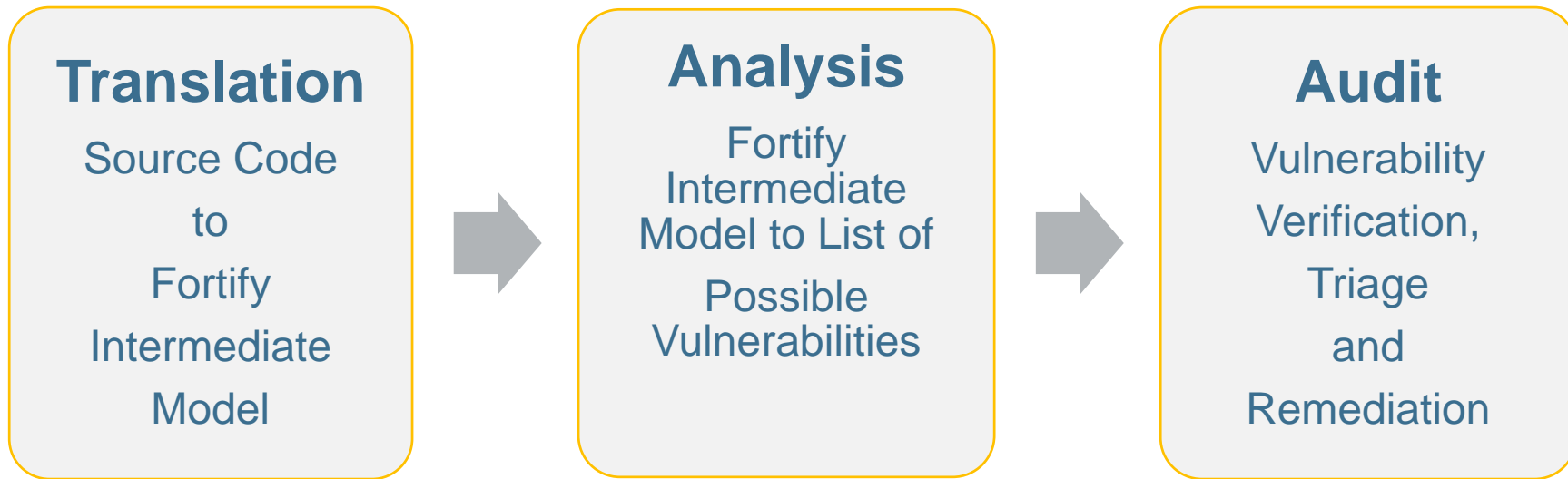
Features:

- Automate static application security testing to identify security vulnerabilities in application source code during development
- Pinpoint the root cause of vulnerabilities with line of code details and remediation guidance
- Prioritize all application vulnerabilities by severity and importance
- Supports 22 languages, 880 vulnerability categories, 806,000 APIs

Benefits:

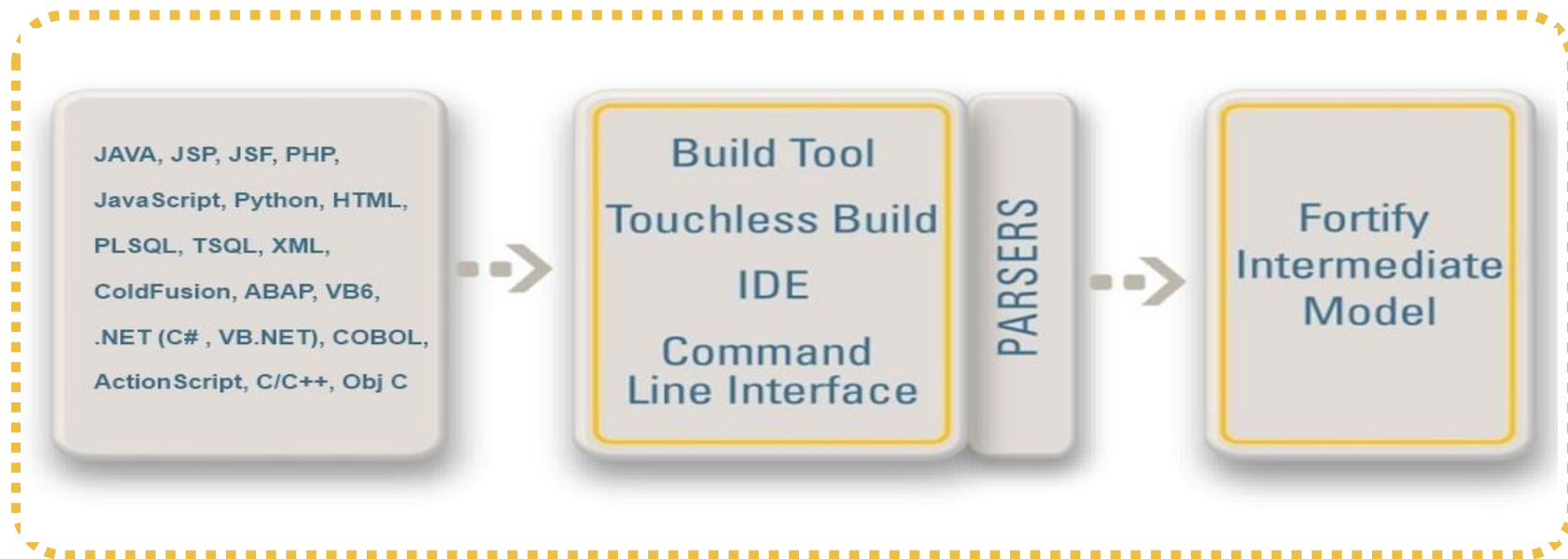
- Reduces the cost of identifying and fixing vulnerabilities
- Reduces risk that a vulnerability will slip by and cause a problem later
- Saves valuable development time and effort

HPE Fortify SCA Process Flow



HPE Fortify SCA Process Flow

Translation Phase



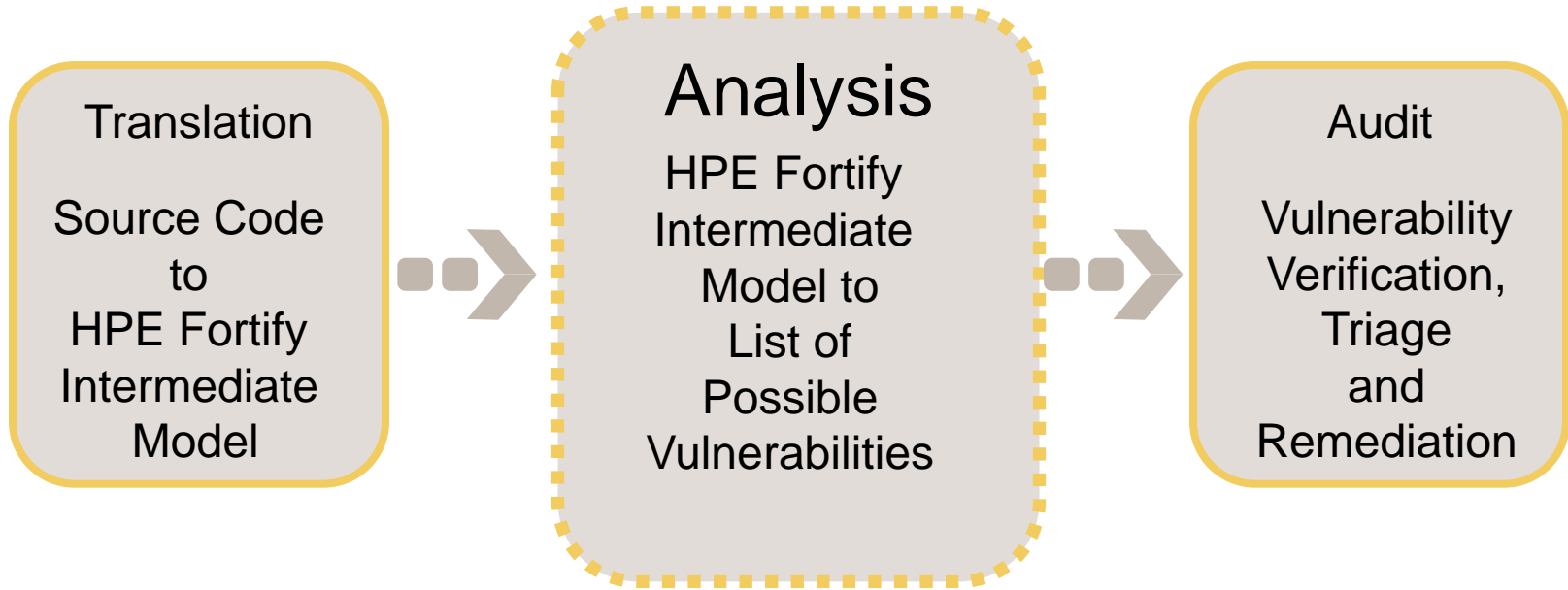
HPE Fortify SCA Process Flow

Translation requirements

–All Languages

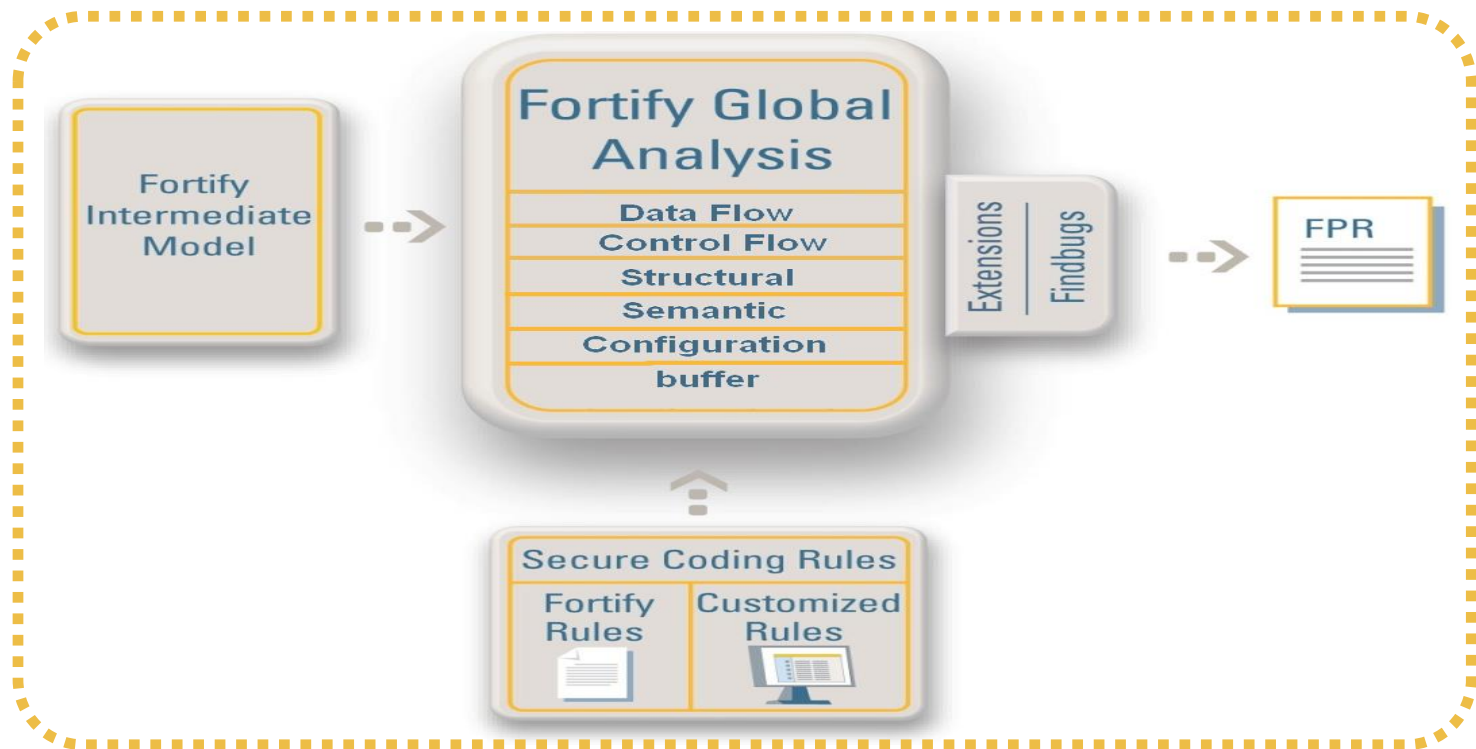
- Source code needs to be in a buildable state
- C, C++, Objective C
 - compiler is required to run SCA translator
- C#, VB.Net
 - solutions must be compiled to create pdb files
- Java, JavaScript, SQL, PHP, ColdFusion, XML, . . .
 - translated directly by the SCA translator

HPE Fortify SCA Process Flow



HPE Fortify SCA Process Flow

Analysis Phase



Secure Coding Rules – From HPE Fortify

–HPE Fortify Rules cover commonly use API

- Library that comes with the programming language
- Common 3rd party and extension library for the language

–Developed by HPE Fortify Security Research Group

- New rule update every quarter
 - Traditionally End of February, May, August, and November
- Distributed as encrypted files
 - HPE Fortify's intellectual property

HPE Fortify SCA Analyzers

Data Flow

- non-trusted input can potentially control application operation.

Control Flow

- Detects potentially dangerous execution sequences

Structural

- Detects potentially dangerous flaws in the structure or definition of a program

Semantic

- Looks for unsafe function calls based on their signature

Configuration

- Uses XPath queries and name-value matching to identify issues in application's configuration files

Buffer

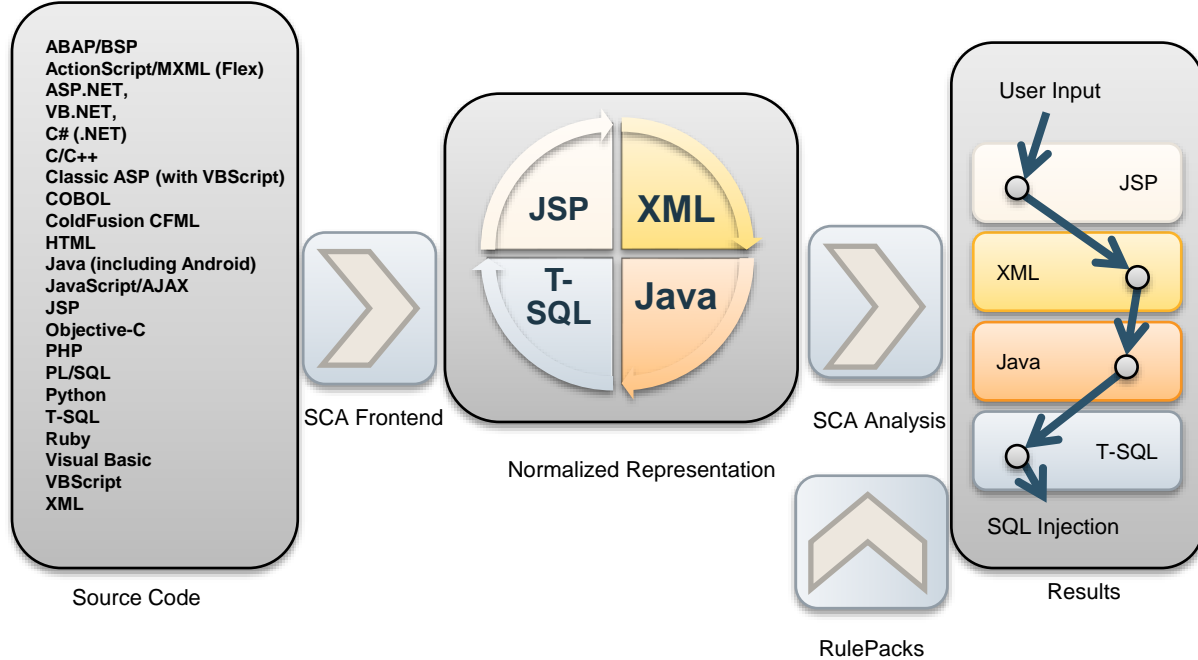
- buffer analyzer detect access to buffer beyond its boundaries

Content

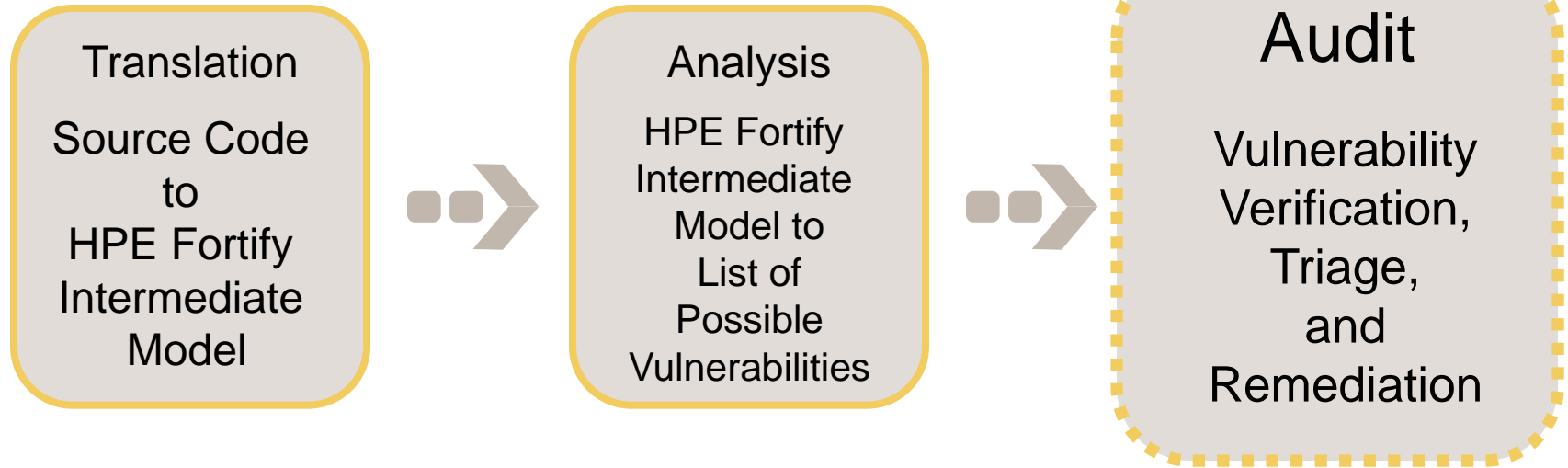
- Searches for security issues and policy violations in HTML

Static Application Security Testing

Accurately identify root cause and remediate underlying security flaw

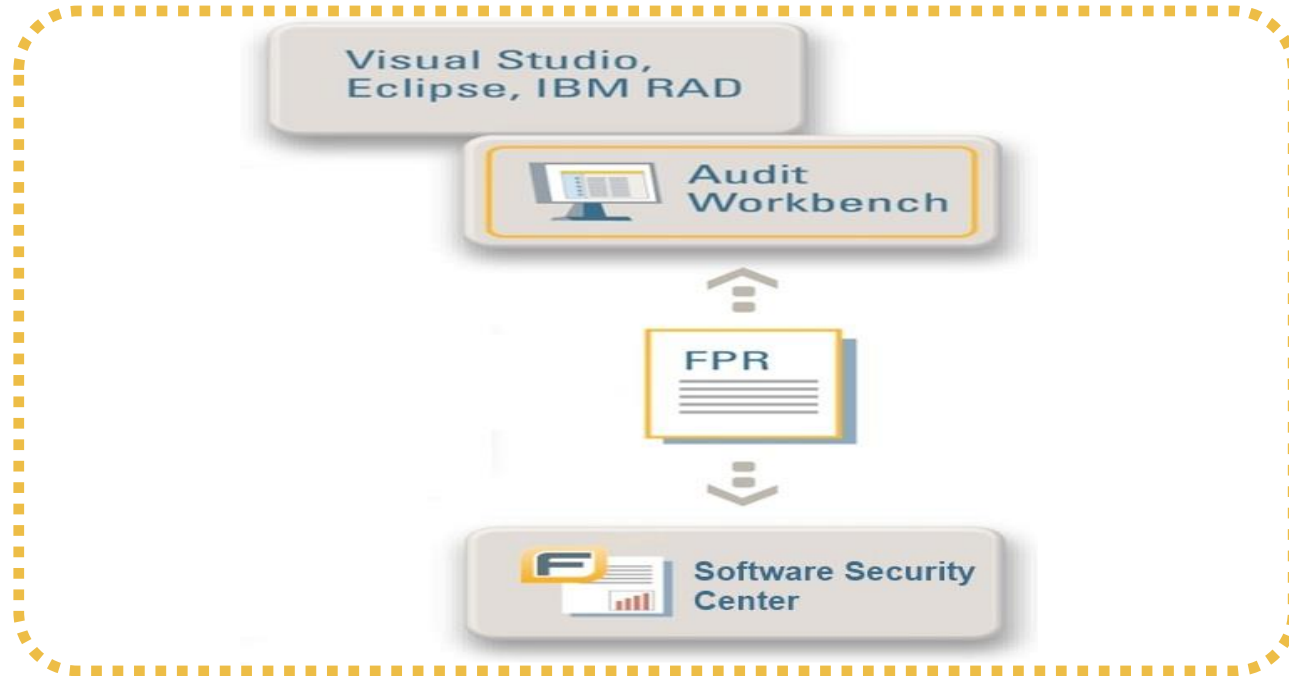


HPE Fortify SCA Process Flow



HPE Fortify SCA Process Flow

Audit Phase

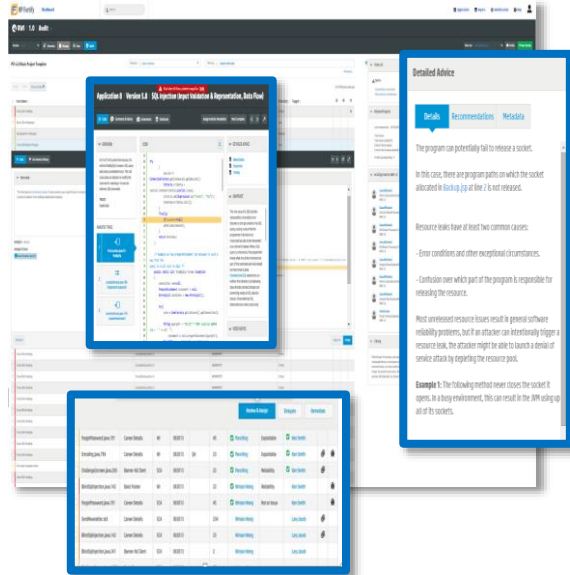


Audit Phase = HPE Fortify Utilities + You

- HPE Fortify Utilities
 - Audit Workbench (AWB)
 - An HPE Fortify Graphical User Interface (GUI) utility
 - Rich features to review results from HPE Fortify SCA analysis
 - Secure Coding Plug-ins
 - Very similar functionalities to AWB
 - Eclipse, Visual Studio
 - Software Security Center (SSC)
 - Contains a collaboration module with similar functionalities to AWB
 - Has a rich reporting interface and stores all findings in the DB
- You
 - Verify and remediate issues found by HPE Fortify SCA

HPE Fortify Software Security Center

Management, tracking and remediation of enterprise software risk



Problem it solves:

Provides visibility into security activities within development

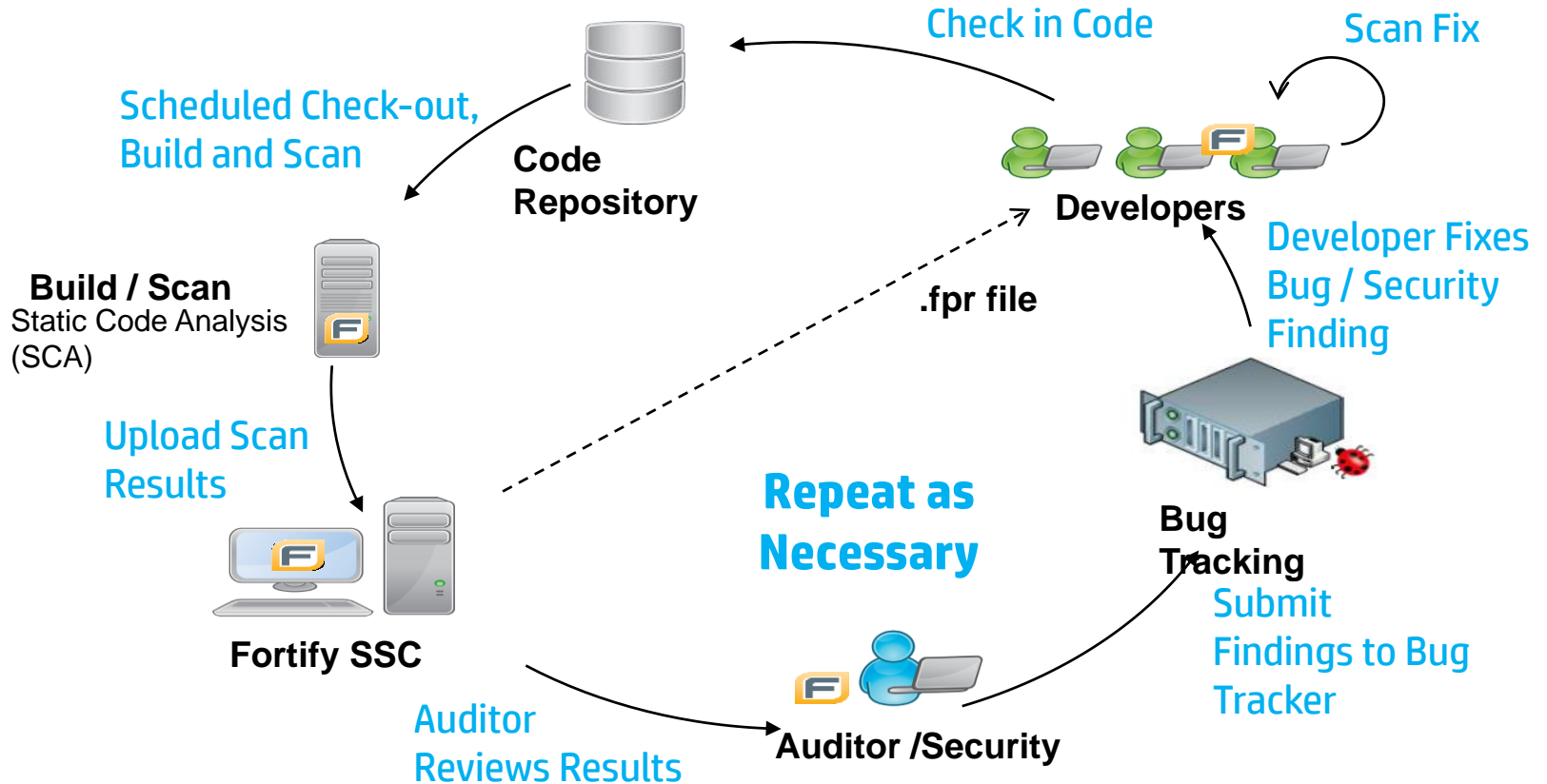
Features:

- Specify, communicate and track security activities performed on projects
- Role-based, process-driven management of software security program
- Flexible repository and exporting platform for security status, trending and compliance

Benefits:

- Provides a clear, accurate picture of software risk across the enterprise
- Lowers cost of resolving vulnerabilities
- Identify areas of improvement for accelerated reduction of risk and costs

Static Software Scanning Process



Fortify SCA Suite

Fortify SCA

Audit WorkBench (AWB)

Secure Coding Plug-ins

Software Security Center

- Static source code analysis.
- Visual interface for analysis of software vulnerabilities.
- Integrated vulnerability detection into Integrated Development Environments (IDEs).
- Management for multiple audit projects from a single centralized console.



How to run a Scan

Fortify scanning

Multiple ways to scan a project

- IDE Plug-ins
- Build integration (Ant, maven, make, Jenkins,...)
- Command Line
- AWB
- Scan Wizard



Using the SCA Command Line interface

SCA Command Line Interface

- Command Line Interface Help

```
sourceanalyzer -h
```

- Java Command Line Syntax

```
sourceanalyzer -b <build_id> -cp <classpath> <file_list>
```

- .NET Command Line Syntax

```
sourceanalyzer -vsversion 9.0 -b MyBuild -libdirs  
ProjOne/Lib;ProjTwo/Lib ProjOne/bin/Debug ProjTwo/bin/Debug
```

- C/C++ Command Line Syntax

```
sourceanalyzer -b <build_id> <compiler> [<compiler options>]
```

Exercise 3: Command-Line Scan

cd

C:\Users\Snowdesc\Desktop\Trainer_Materials\riches_wealth_src

CLEAN

```
sourceanalyzer -b riches-class -clean
```

TRANSLATE

```
sourceanalyzer -b riches-class -sql-language PL/SQL -source  
1.6 -cp ./WEB-INF/lib/*.jar;./lib/*.jar ./**/*.java ./**/*.jsp  
./**/*.sql ./**/*.xml ./**/*.js ./**/*.html
```

SCAN

```
sourceanalyzer -b riches-class -source 1.6 -Xmx3200M -64 -  
scan -f richesresults.fpr
```





Using the Eclipse Plugin

HPE Fortify SCA – Eclipse IDE Plug-in

The screenshot displays the Eclipse IDE interface with the HPE Fortify SCA plug-in. The top menu bar includes 'File', 'Edit', 'Source', 'Refactor', 'Navigate', 'Search', 'Project', 'Run', 'HP Fortify', and 'Window'. The 'HP Fortify' menu is circled in red. The main workspace shows a 'Project Summary' view for a project named 'riches44'. The summary includes:

- Project Location: C:\Apps\riches\riches44.fpr
- Scanned: 161 files, 2,510 LOC (Executable)
- Project Version: riches44
- Total Issues: 376
- Scan Date: Nov 23, 2015
- Total LOC: 4,314
- Warnings: 15 occurred during scan
- Certification: Results Certification Valid

Below the summary is a bar chart titled 'All Issues by Folder' showing the distribution of issues by severity:

Severity	Count
Medium	7
Critical	56
High	71
Low	242
All	376

The right-hand pane shows 'SCA Analysis Results' with a filter set to 'Security Auditor \\' and 'My Issues'. It displays a summary of 'Critical (56) Suppressed (0)' issues and a list of categories with their respective counts:

- Command Injection - [0 / 1]
- Cross-Site Scripting: DOM - [0 / 4]
- Cross-Site Scripting: Persistent - [0 / 4]
- Cross-Site Scripting: Reflected - [0 / 8]
- Dangerous File Inclusion - [0 / 2]
- Open Redirect - [0 / 1]
- Path Manipulation - [0 / 2]
- Privacy Violation - [0 / 15]
- SQL Injection - [0 / 4]
- SQL Injection: Hibernate - [0 / 13]
- XML External Entity Injection - [0 / 2]

The bottom of the IDE shows a 'Recommendations' tab and a 'Webinspect Agent Details' tab. The bottom-left corner features the Hewlett Enterprise logo.

HPE Fortify SCA – Eclipse IDE Plug-in

The screenshot displays the Eclipse IDE interface with the HPE Fortify SCA plug-in. The Package Explorer on the left shows a project named 'Riches' with a 'java' package containing several sub-packages like 'com.fortify.samples.riches'. The main editor shows the 'LocationService.java' file with Java code. A 'Scanning' dialog box is open, indicating that Fortify SCA is scanning the code. The dialog includes a progress bar and a checkbox for 'Always run in background'. The bottom of the IDE shows a toolbar with various analysis tools and a status bar indicating 'Scanning: (10%)'.

```
116         *           ResultSet rs= prepStmt.executeQuery();/*
117
118
119
120     String queryStr = "SELECT * FROM location WHERE zip = '" + zip + "'";
121     statement = conn.prepareStatement(queryStr);
122     ResultSet rs = statement.executeQuery();
123     while (rs.next())
124     {
125         locations.add(new Location(rs.getString("address"),rs.getString("city"),rs.getString("state"),rs.getString("zip"),rs.getString("atm"),rs.getString("branch"));
126     }
127 }
128 finally{
129     safeCloseStatement(statement);
130     safeCloseConnecti(conn);
131 }
132
133 return locations;
134
135 }
136
137
138
139
140 public static List findAtms(String zip, String state, String city, String address)
141 {
142     Connection conn=null;
143     Statement statement= null;
144     ArrayList locations = new ArrayList();
145
146     try{
147         conn = ConnFactory.getInstance().getConnection();
148         String queryStr = "SELECT * FROM location WHERE branch = 'Yes' AND state = '" + state + "' AND city = '" + city + "' AND address = '" + address + "'";
```


Fortify SCA – Eclipse IDE Plug-in

The screenshot shows the Eclipse IDE interface with the Fortify SCA plug-in. The main editor displays the `LocationService.java` file, which contains a SQL query: `"SELECT * FROM location WHERE zip = '" + zip + "'";`. The SCA Analysis Results panel on the right shows a list of 56 critical issues, including Command Injection, Cross-Site Scripting, Dangerous File Inclusion, Open Redirect, Path Manipulation, Privacy Violation, SQL Injection, and XML External Entity Injection.

```
116      *      ResultSet rs= prepStmt.executeQuery();/*
117
118
119
120      String queryStr = "SELECT * FROM location WHERE zip = '" + zip + "'";
121      statement = conn.prepareStatement(queryStr);
122      ResultSet rs = statement.executeQuery();
123      while (rs.next())
124      {
125          locations.add(new Location(rs.getString("address"), rs.getString("city"), rs.getString("state"), rs.getString("zip"));
126      }
127
128      finally{
129          safeCloseStatement(statement);
130          safeCloseConnection(conn);
131      }
132
133      return locations;
134
135
136
137
138
139      public static List findAtmByAddress(String address, String city, String state) throws Exception
140      {
141          Connection conn=null;
142          Statement statement = null;
143          ArrayList locations = new ArrayList();
144
145          try{
146              conn = ConnFactory.getInstance().getConnection();
147
148              String queryStr = "SELECT * FROM location WHERE branch = 'Yes' AND state = '" + state + "' AND city = '" + city + "' AND addr
```

Noted: you should make sure all libraries are included, and source codes are compliable before you scan.

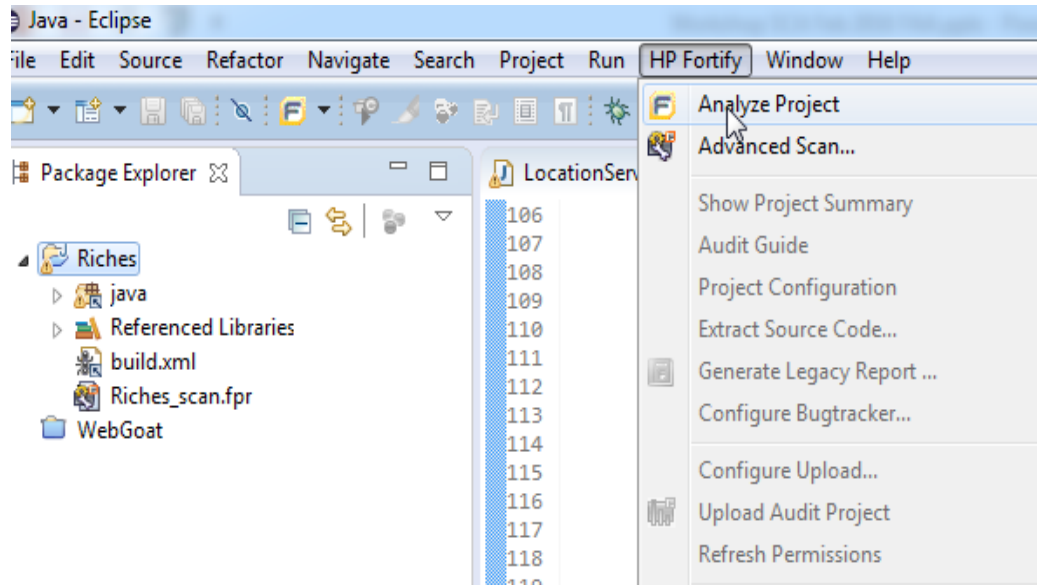
HPE Fortify SCA – Eclipse IDE Plug-in

The screenshot displays the Eclipse IDE interface with the HPE Fortify SCA plug-in. The interface is divided into several views:

- Package Explorer:** Located on the left, it shows a project structure with folders for 'Java' and 'Referenced Libraries'. A file named 'riches_scan.fpr.scan' is selected.
- Source Code:** The main editor area shows a 'Project Summary' view. It displays scan statistics: Scanned: 160 files, 2,625 LOC (Executable); Total Issues: 391; Total LOC: 4,478. Below this is a bar chart titled 'All Issues by Folder' showing the distribution of issues by severity: Critical (59), High (79), and Low (248).
- Analysis Result:** Located on the right, it shows a list of detected issues. The top section indicates 'Critical (59) Suppressed (0)'. Below this is a list of issue categories with their counts: Cross-Site Scripting: DOM - [0 / 4], Cross-Site Scripting: Persistent - [0 / 5], Cross-Site Scripting: Reflected - [0 / 11], Dangerous File Inclusion - [0 / 2], Open Redirect - [0 / 1], Path Manipulation - [0 / 2], Privacy Violation - [0 / 11], SQL Injection - [0 / 4], SQL Injection: Hibernate - [0 / 13], XML Entity Expansion Injection - [0 / 2], and XML External Entity Injection - [0 / 2].
- Summary and details:** Located at the bottom left, it shows a detailed view of an issue. It includes tabs for 'Analysis Evidence', 'Diagram', 'History', 'Issue Details', 'Issue Summary', and 'Recommendations'. The 'Issue Summary' tab is active, showing fields for 'Issue:', 'User:', and 'Analysis:'. There is also a 'Show merge conflicts' checkbox.
- Analysis Trace:** Located at the bottom right, it shows a detailed view of the analysis process. It includes a search bar and a 'Recommendations...' button.

Exercise 4: Eclipse IDE Plugin Scan

- In Package Explorer → Open Project Riches
- HP Fortify → Analyze Project
- Start Scan





Remediate/Rescan

Exercise 5: Remediate SQLI and Rescan

The screenshot displays a Java code editor on the left and a Security Auditor View window on the right. The code in the editor is as follows:

```
36     try{
37         conn = ConnFactory.getInstance().getConnection();
38
39         /*this is the FIX for the SQL INJECTION for LCOATIONJAVA.110
40
41         *         String selectStatement = "SELECT * FROM location WHERE zip = ?";
42         *         PreparedStatement prepStmt = con.prepareStatement(selectStatement);
43         *         preparedStatement.setString(1, zip);
44         *         ResultSet rs= prepStmt.executeQuery();*/
45
46         String queryStr = "SELECT * FROM location WHERE zip = '" + zip + "'";
47         statement = conn.prepareStatement(queryStr);
48         ResultSet rs = statement.executeQuery();
49         while (rs.next())
50         {
51             locations.add(new Location(rs.getString("address"), rs.getString("city"), rs.getString("state"), rs.get
52         }
53     }
54     finally{
55         safeCloseStatement(statement);
56         safeCloseConnection(conn);
57     }
58     return locations;
59 }
```

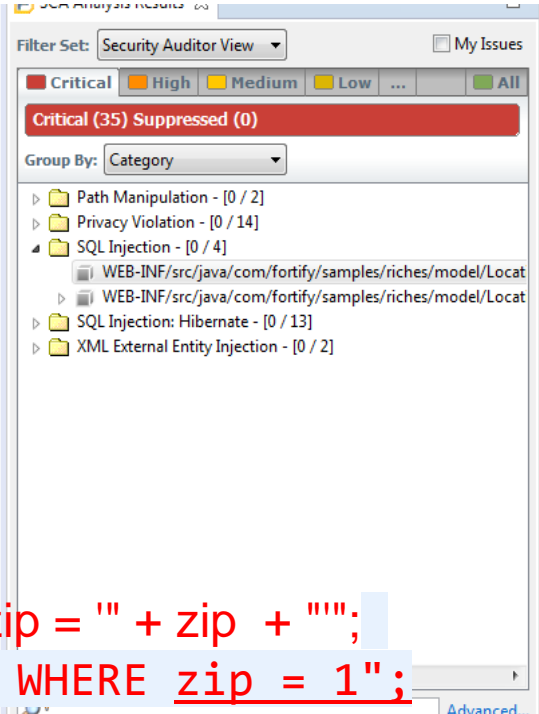
The line `String queryStr = "SELECT * FROM location WHERE zip = '" + zip + "'";` is circled in red, indicating a SQL injection vulnerability. The Security Auditor View window on the right shows the following details:

- Filter Set: Security Auditor View
- My Issues:
- Critical (35) Suppressed (0)
- Group By: Category
- Issues List:
 - Path Manipulation - [0 / 2]
 - Privacy Violation - [0 / 14]
 - SQL Injection - [0 / 4]
 - WEB-INF/src/java/com/fortify/samples/riches/model/Locat
 - WEB-INF/src/java/com/fortify/samples/riches/model/Locat
 - SQL Injection: Hibernate - [0 / 13]
 - XML External Entity Injection - [0 / 2]

Exercise 5: Remediate SQLI and Rescan

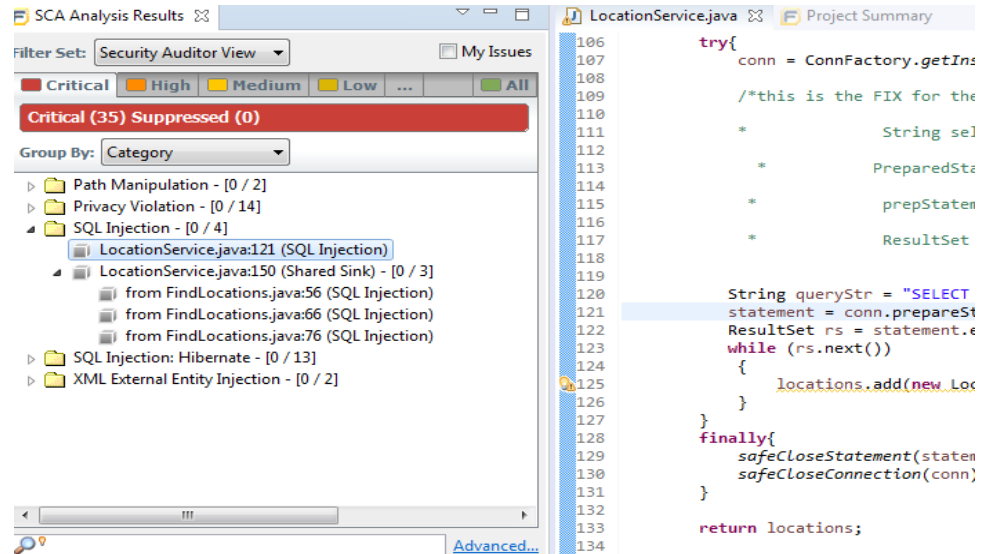
```
36     try{
37         conn = ConnFactory.getInstance().getConnection();
38
39         /*this is the FIX for the SQL INJECTION for LCOATIONJAVA.110
40
41         *         String selectStatement = "SELECT * FROM location WHERE zip = ?";
42
43         *         PreparedStatement prepStmt = con.prepareStatement(selectStatement);
44
45         *         preparedStatement.setString(1, zip);
46
47         *         ResultSet rs= prepStmt.executeQuery();*/
48
49
50         String queryStr = "SELECT * FROM location WHERE zip = " + zip + """;
51         statement = conn.prepareStatement(queryStr);
52         ResultSet rs = statement.executeQuery();
53         while (rs.next())
54         {
55             locations.add(new Location(rs.getString("address"), rs.getString("city"), rs.getString("state"), rs.get
56         }
57     }
58     finally{
59         safeCloseStatement(statement);
60         safeCloseConnection(conn);
61     }
62
63     return locations;
64 }
```

String queryStr = "SELECT * FROM location WHERE zip = " + zip + """;
// String queryStr = "SELECT * FROM location WHERE zip = 1";



Exercise 6: Remediate SQLI and Rescan

- SCA Analysis Result → Find SQL Injection
- Expand SQL Injection → Choose LocationService.Java:121
- Determine if the SQLI is exploitable or not
- Make change to the code
- Rescan

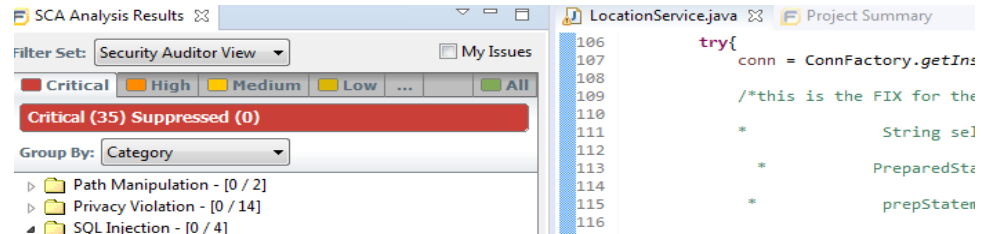


The screenshot shows the SCA Analysis Results window on the left and the LocationService.java code editor on the right. The SCA window displays a tree view of issues, with 'SQL Injection - [0 / 4]' expanded to show 'LocationService.java:121 (SQL Injection)'. The code editor shows the following Java code:

```
106     try{
107         conn = ConnFactory.getIn
108
109         /*this is the FIX for the
110         *
111         *         String sel
112         *
113         *         PreparedStatement
114
115         *         preparedStatement
116         *
117         *         ResultSet
118
119
120         String queryStr = "SELECT
121         statement = conn.prepareStatement
122         ResultSet rs = statement.e
123         while (rs.next())
124         {
125             locations.add(new Loc
126         }
127     }
128     finally{
129         safeCloseStatement(statement)
130         safeCloseConnection(conn)
131     }
132
133     return locations;
134
```

Exercise 6: Remediate SQLI and Rescan

- SCA Analysis Result → Find SQL Injection
- Expand SQL Injection → Choose LocationService.Java:121
- Determine if the SQLI is exploitable or not
- Make change to the code
- Rescan



```
String queryStr = "SELECT * FROM location WHERE zip = ?";  
//String queryStr = "SELECT * FROM location WHERE zip = " + zip + "";  
// String queryStr = "SELECT * FROM location WHERE zip = 1";  
statement = conn.prepareStatement(queryStr);  
statement.setString(1, zip);
```

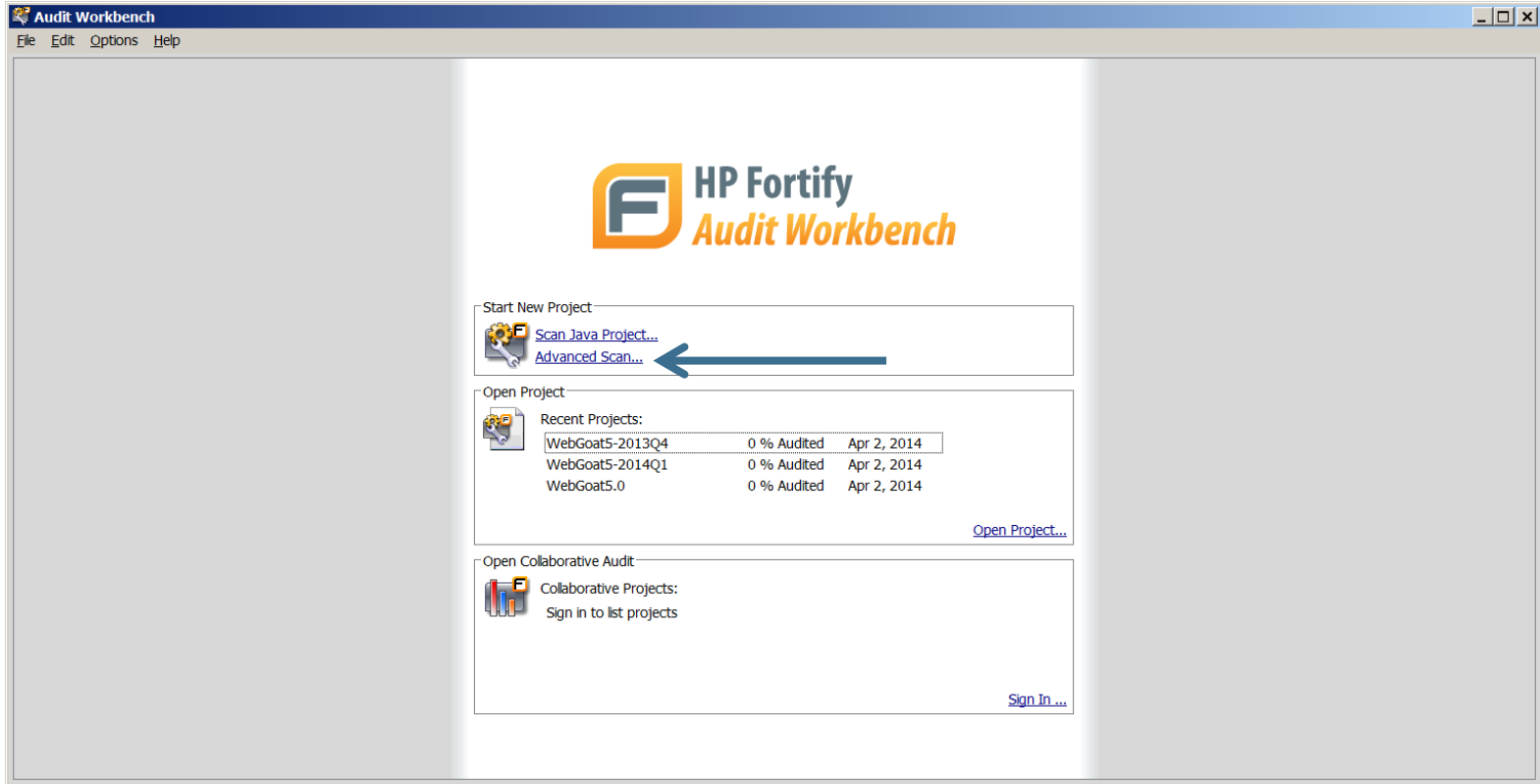
```
String queryStr = "SELECT * FROM location WHERE zip = ?";  
statement = conn.prepareStatement(queryStr);  
ResultSet rs = statement.executeQuery(queryStr);  
while (rs.next())  
{  
    locations.add(new Location(rs.getString("location"), rs.getString("zip")));  
}  
finally{  
    safeCloseStatement(statement);  
    safeCloseConnection(conn);  
}  
return locations;
```



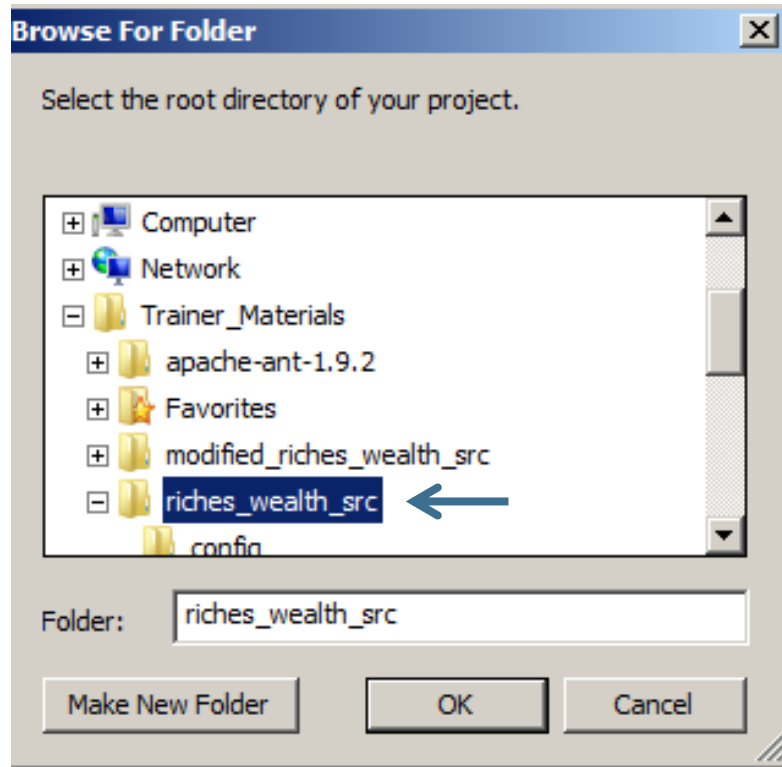


Using the AWB

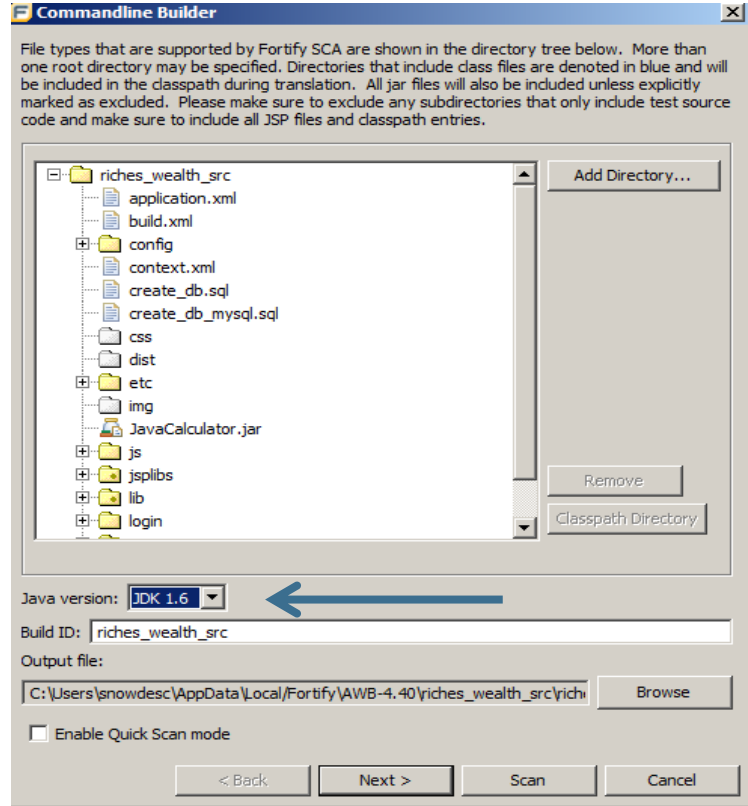
Exercise 7: Scanning With Audit Workbench



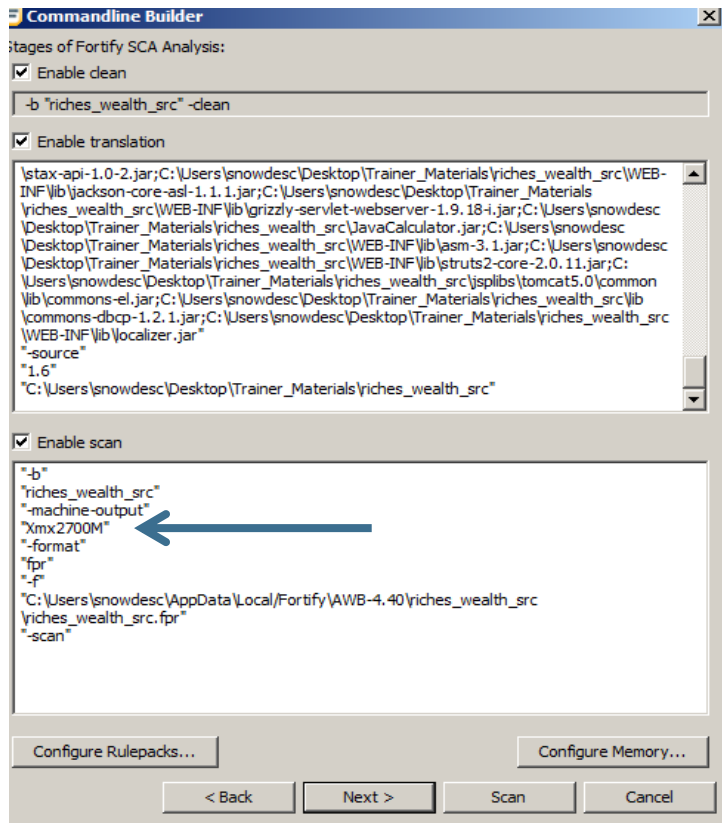
Exercise 7: Scanning With Audit WorkBench



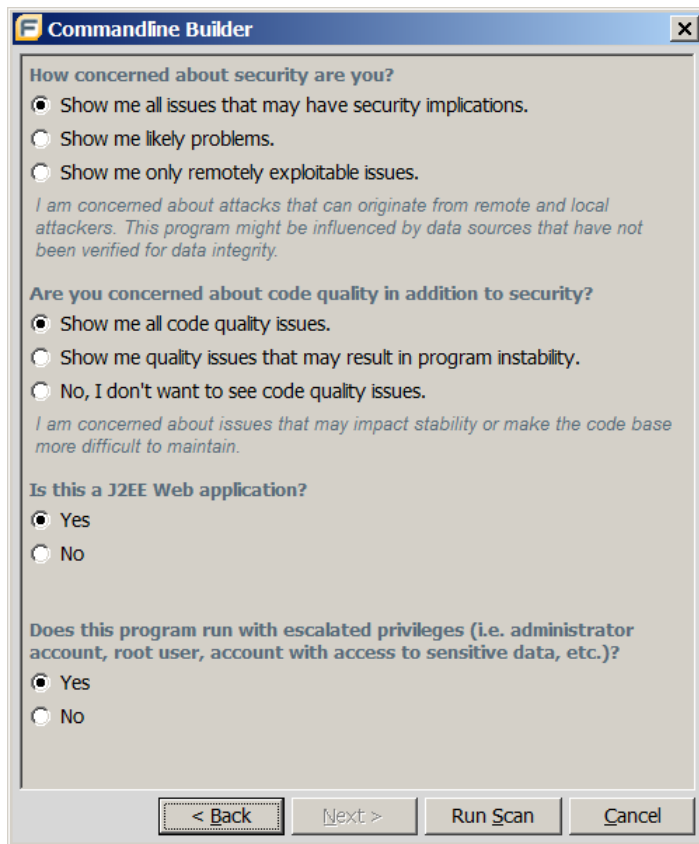
Exercise 7: Scanning With Audit Workbench



Exercise 7: Scanning With Audit Workbench



Exercise 7: Scanning With Audit Workbench



Commandline Builder

How concerned about security are you?

- Show me all issues that may have security implications.
- Show me likely problems.
- Show me only remotely exploitable issues.

I am concerned about attacks that can originate from remote and local attackers. This program might be influenced by data sources that have not been verified for data integrity.

Are you concerned about code quality in addition to security?

- Show me all code quality issues.
- Show me quality issues that may result in program instability.
- No, I don't want to see code quality issues.

I am concerned about issues that may impact stability or make the code base more difficult to maintain.

Is this a J2EE Web application?

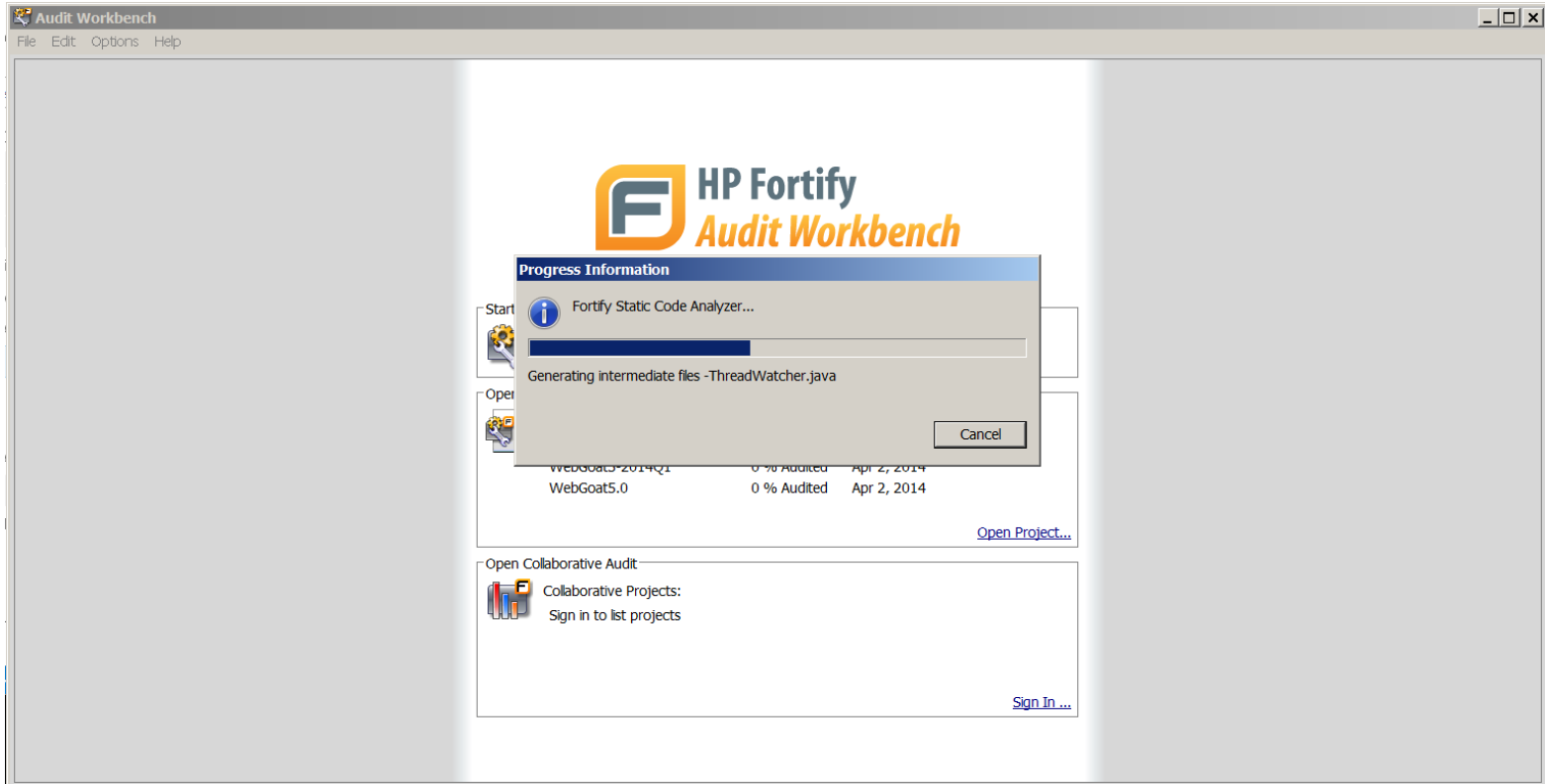
- Yes
- No

Does this program run with escalated privileges (i.e. administrator account, root user, account with access to sensitive data, etc.)?

- Yes
- No

< Back Next > Run Scan Cancel

Exercise 7: Scanning With Audit Workbench



Audit Workbench Scan Exercise

- Start Audit Workbench
- Select “Advanced Scan...”
- Navigate to
`C:\Users\SnowDesc\Desktop\Trainer_Materials\riches_wealth_src`
- Click **OK**
- Specify Java Version 1.6
- Click **Next >**
- Add “-Xmx2700M” to translation and scan command line options
- Click **Next >** then click **Scan**



Configuring AWB

Configuration – Options



Server Configuration – where to ...

Options...

Server Configuration

Server Configuration
Security Content Management
Interface Preferences
Audit Features Configuration

Security Content Update Configuration

Server URL:
Proxy Server: Port:
 Perform Security Content Update Analysis
Security Content Update Path:

Software Security Center Configuration

Server URL:
Port:
Username:
Password:

Audit Workbench Upgrade Configuration

Server URL:
Check Now
Defaults

Using HP Fortify Static Code Analyzer **6.40.0089** using JVM 1.8.0_45

OK Cancel

Where to DOWNLOAD rulepacks

Where to UPLOAD scan results

Where to get SCA updates

SCA Version

Server Configuration – details

Options...

Server Configuration

Security Content Update Configuration

Update Security Content from Software Security Center

Server URL:

Proxy Server: Port:

Perform Security Content Update Automatically

Security Content Update Frequency (Days)

Software Security Center Configuration

Server URL:

Proxy Server: Port:

Audit Workbench Upgrade Configuration

Server URL:

Check for upgrades at startup

Check Now

Defaults

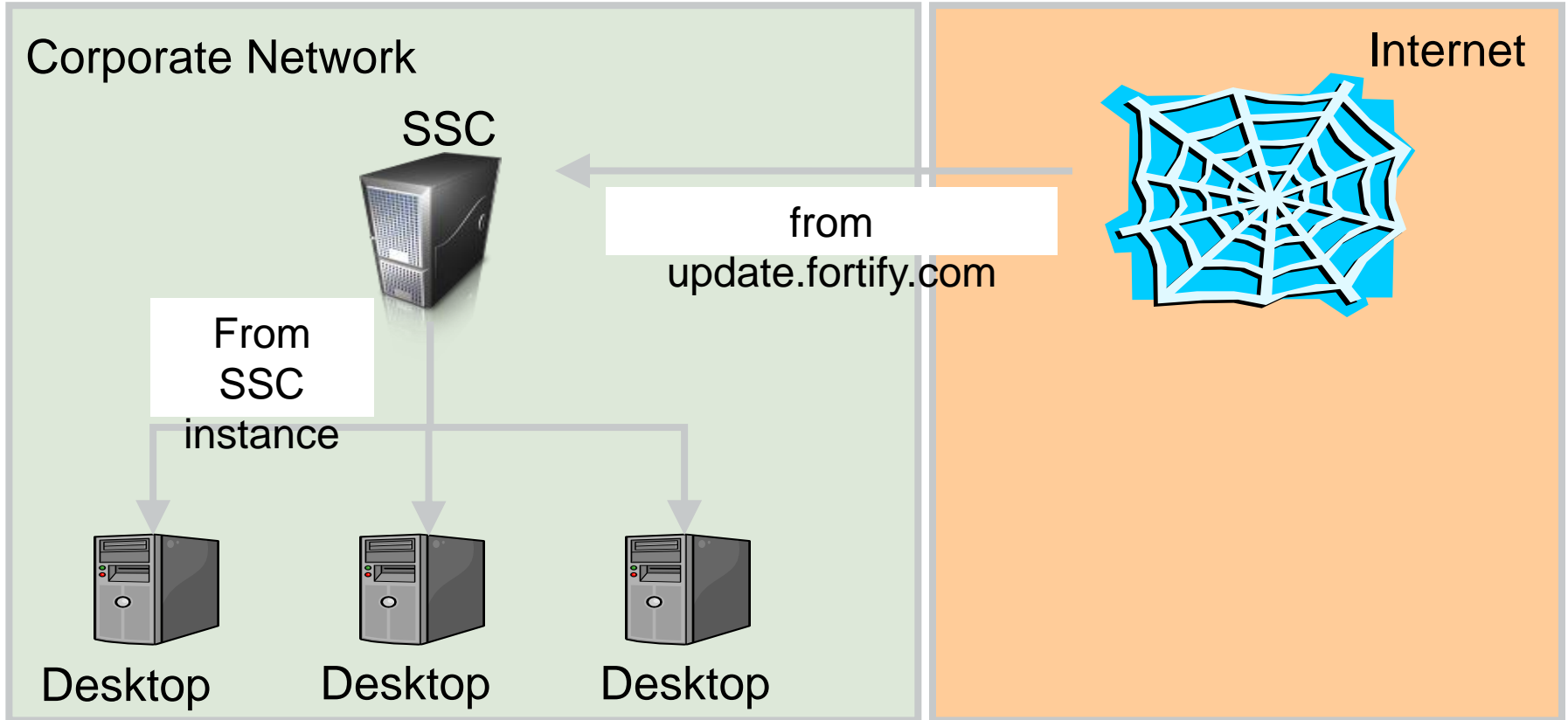
OK Cancel

Using Fortify Static Code Analyzer 6.21.0007

If you have an SSC instance then rulepacks can be downloaded from it.

Can automatically check for rulepack updates.

Typical Configuration – download rulepack



Typical Configuration – upload scan results

Corporate Network

SSC



Upload scan
results to SSC
instance



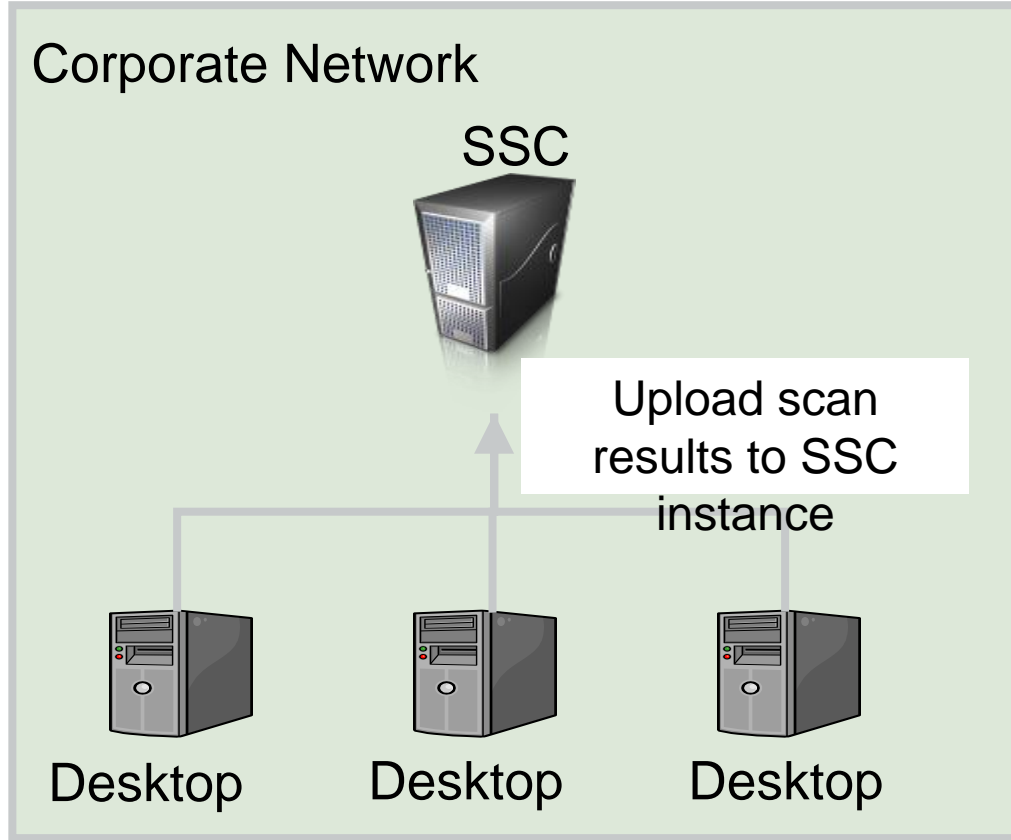
Desktop



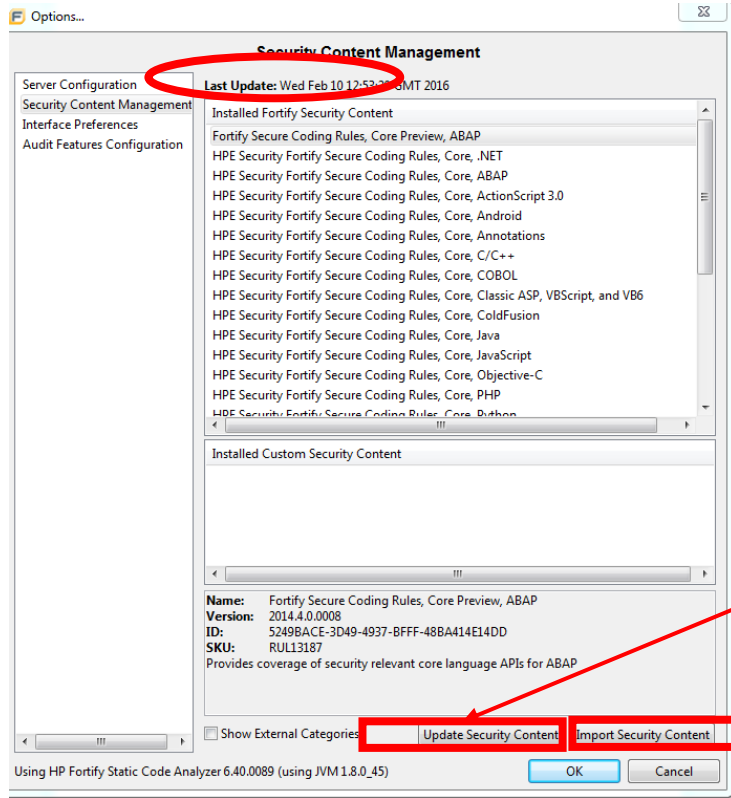
Desktop



Desktop



Security Content Management

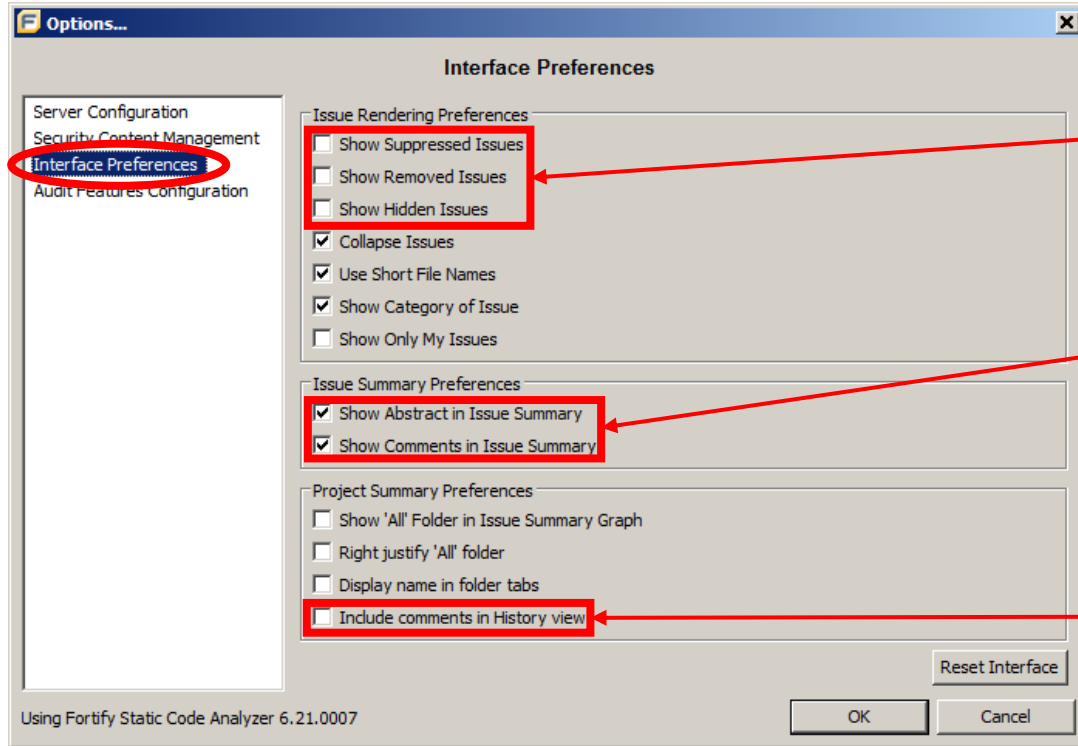


Security
Content Version

Click to
manually check
for new security
content

Click to import
security content
(custom rules)

Interface Preferences

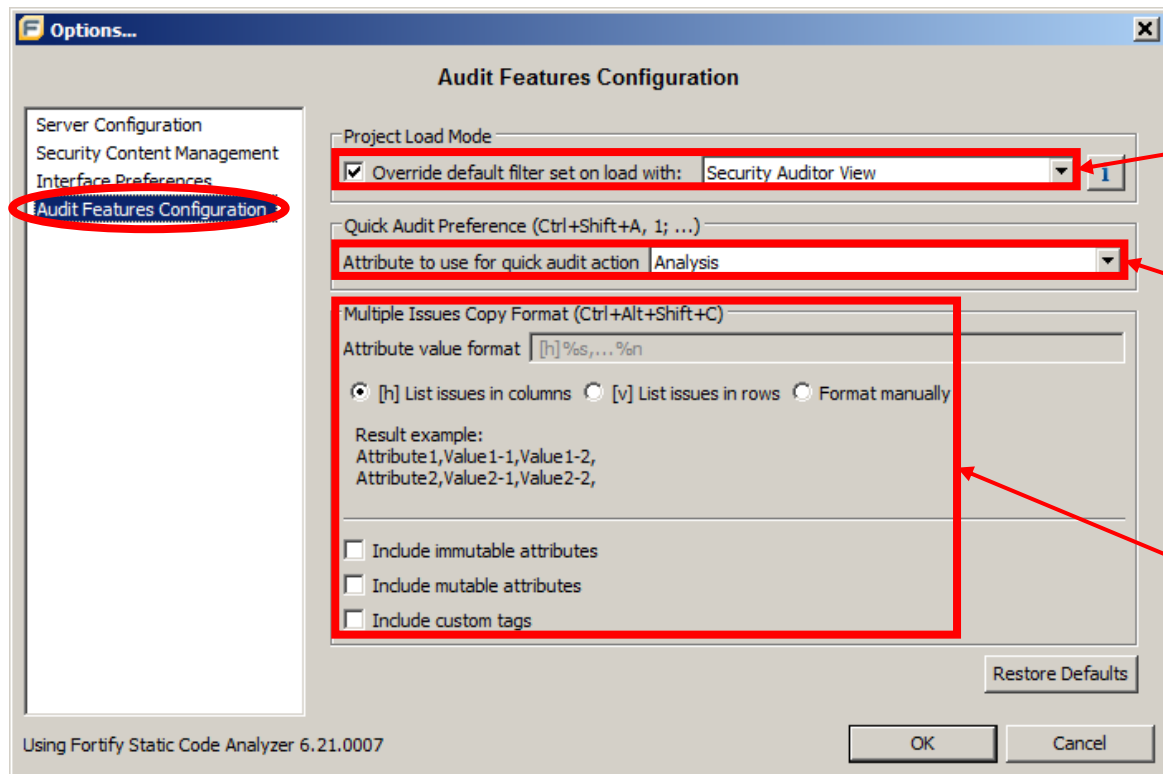


Control Issue Display

Tweak Issue Summary Display

Merge comments with analysis tag

Audit Features Configuration



Override default
Filter Set

Set attribute to
use for quick
audit feature

Configure how
issues are
copied

Memory Considerations

–SCA is a java application

Can be set using the `-Xmx` command line option

`-Xmx10800M`

`-Xmx8G`

–You can set the maximum java heap space via environment variable

`SCA_VM_OPTS=-Xmx4G` –used for SCA

`AWB_VM_OPTS=-Xmx2G` –used for Audit Workbench

–Value should not be greater than two thirds of total system memory



Working with the Results

Working with the results . . .

The screenshot displays the Fortify Audit Workbench interface for a project named 'WebGoat5.0'. The main window is divided into several panes:

- Issues (Left Panel):** A tree view showing 258 critical issues. The selected issue is 'Cross-Site Scripting: Persistent - [0 / 38]' located in 'BackDoors.java:126 (Cross-Site Scripting: Persistent)'. Other issues include CSRF vulnerabilities and a shared sink in DatabaseUtilities.java.
- Source Code (Center Panel):** A code editor showing the source code for 'BackDoors.java'. Lines 122-136 are visible, showing a loop that adds elements to a table. Line 126 is highlighted, corresponding to the selected issue: `tr.addElement(new TD(rs.getString("password")));`
- Issue Auditing (Bottom Center Panel):** A detailed view of the selected issue. It shows the user, analysis, and a description: 'Cross-Site Scripting: Persistent (Input Validation and Representation, Data flow). The method concept1() in BackDoors.java sends unvalidated data to a web browser on line 126, which can result in the browser executing malicious code.' It also includes a 'More Information...' link and a 'Recommendations...' link.
- Functions (Right Panel):** A list of functions used in the code, including 'Top-level functions', 'java.io', 'java.lang', 'java.sql', 'java.text', 'java.util', 'java.util.regex', 'javax.crypto', 'javax.crypto.spec', 'javax.servlet', 'javax.servlet.http', and 'javax.servlet.jsp'.
- Analysis Evidence (Bottom Left Panel):** A list of analysis steps for the selected issue, including 'executeQuery(return)', 'Assignment to rs', 'getString(this : return)', and 'executeQuery(return)'. The rule ID is '94B3F80E-4AED-4006-9CDD-82B1C13747EE' and the taint flags are 'DATABASE, XSS'.

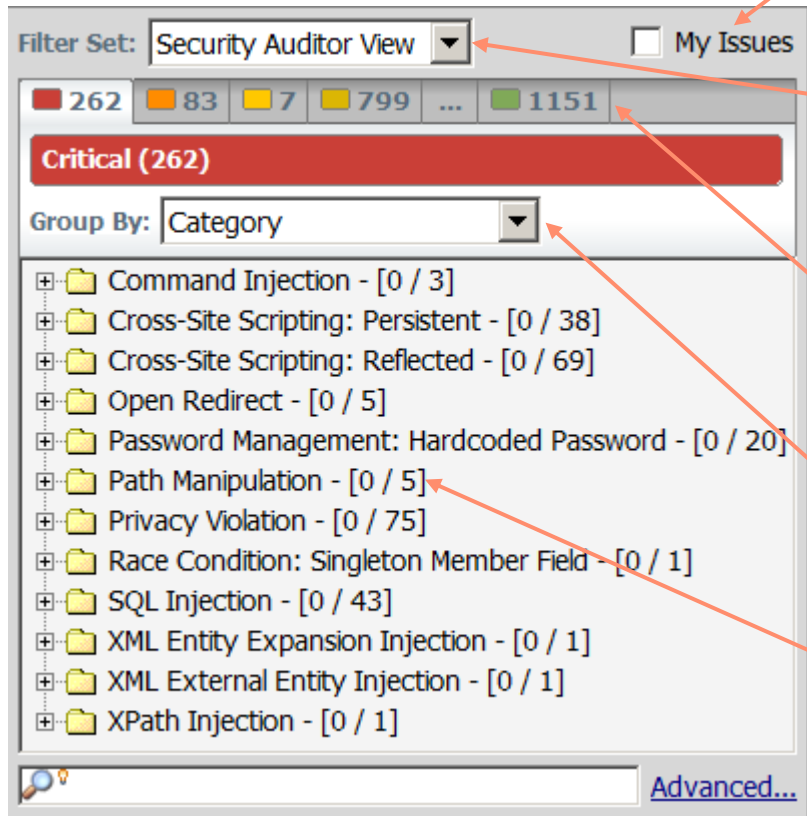
Working with the results . . .

The screenshot displays the Fortify Audit Workbench interface for a Java project. The main window is titled "WebGoat5.0 - JavaSource/org/owasp/webgoat/lessons/BackDoors.java - Audit Workbench". The interface is divided into several panes:

- Issues:** A yellow sidebar on the left shows a list of issues. The "Critical (258)" category is selected, and the "Group By" is set to "Category". A tree view shows "Cross-Site Scripting: Persistent - [0 / 38]" with sub-items for various lines of code in BackDoors.java and CSRF.java.
- Source Code:** The central pane shows the source code for BackDoors.java. Lines 122-136 are visible, showing the addition of elements to a table row. Line 126 is highlighted, corresponding to the selected issue.
- Issue Auditing:** The bottom pane shows the "Issue: BackDoors.java:126 (Cross-Site Scripting: Persistent)". It includes a "User:" field, an "Analysis:" dropdown, and a detailed description of the issue: "Cross-Site Scripting: Persistent (Input Validation and Representation, Data flow). The method concept1() in BackDoors.java sends unvalidated data to a web browser on line 126, which can result in the browser executing malicious code." There are links for "More Information..." and "Recommendations...".
- Analysis Evidence:** The bottom-left pane shows a list of analysis evidence items, including "BackDoors.java:113 - executeQuery(return)", "BackDoors.java:113 - Assignment to rs", "BackDoors.java:126 - getString(this : return)", and "BackDoors.java:126 - getString(this : return)".
- Functions:** A sidebar on the right lists various functions and packages, including "Top-level functions", "java.io", "java.lang", "java.sql", "java.text", "java.util", "java.util.regex", "javax.crypto", "javax.crypto.spec", "javax.servlet", "javax.servlet.http", and "javax.servlet.jsp".

At the bottom left, the text "Enterprise" is visible.

Issues Panel



My Issues: Filter issues to only display issues assigned to current user.

Filter Set: Current Filter being used to display issues.

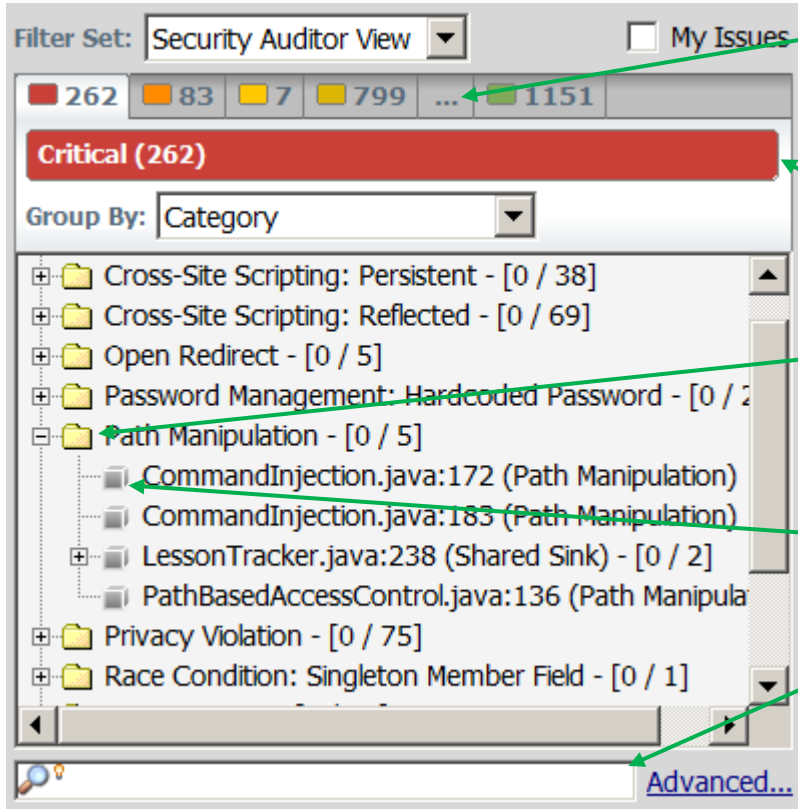
Folders: One tab for each folder. The default folders are one for each Fortify Priority and one for all issues.

Critical, High, Medium, Low, ALL

Group By: How the issues are grouped together. The default grouping is by Category.

Group Counts: This shows how many issues have been audited and how many total issues for the group. 0/5 means a total of 5 issues with 0 (zero) issues audited.

Issues Panel



Folder Dialog Shortcut: Clicking the ... will open the dialog to edit folders.

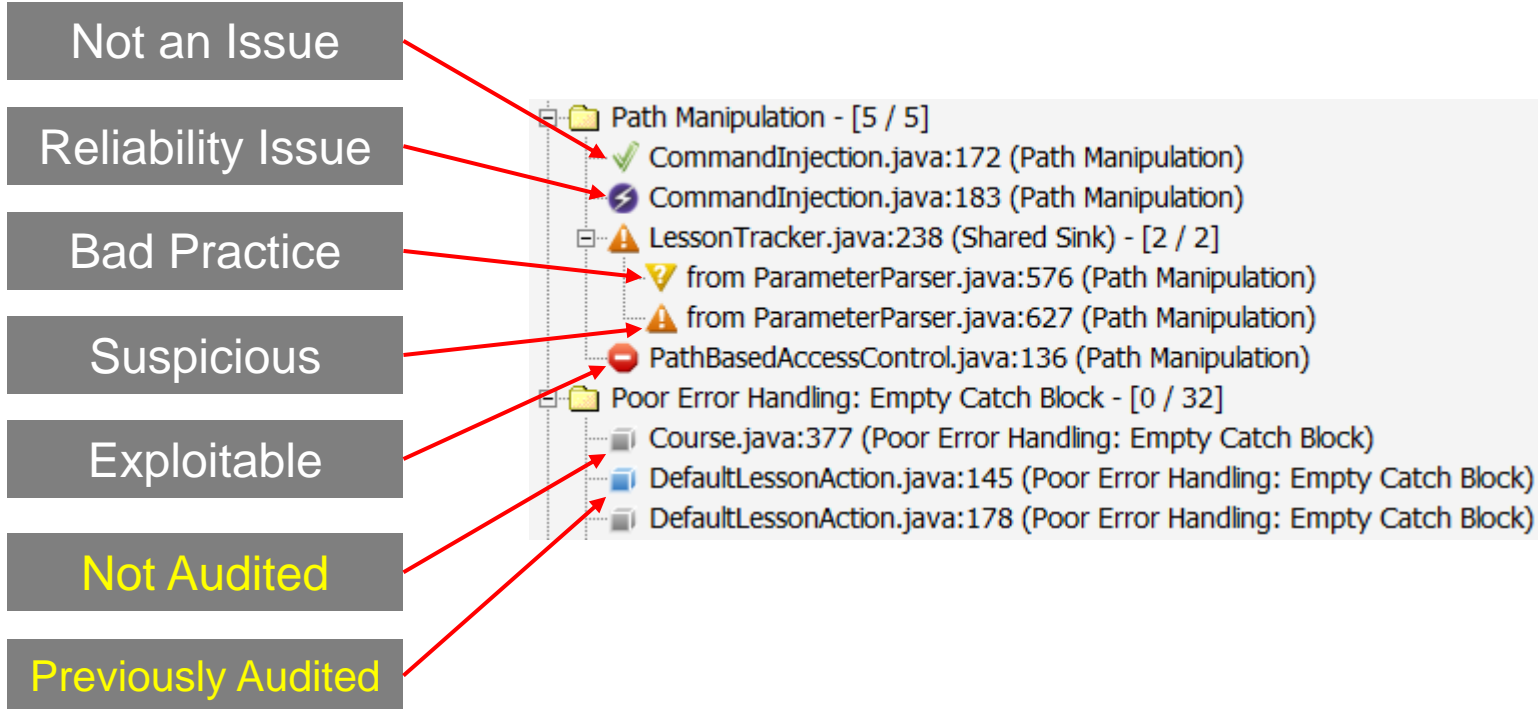
Folder Name: Name of the currently selected folder.

Group Icon: A folder graphic is used as the group icon. This should not be confused with the Folders that are represented by the tabs.

Issue Icon: Represents the auditing that has been done for the issue.

Search: Used to search issues. Advanced searching capabilities are available.

Issues Panel – analysis tag icons



Issue Grouping

Analyzer

- Configuration - [0 / 15]
- content - [0 / 42]
- Control flow - [0 / 53]
- Data flow - [0 / 372]
- Semantic - [0 / 244]
- Structural - [0 / 424]

RTA Protected

- Not Protected - [0 / 32]
- Protected - [0 / 226]

Create Custom Grouping

The screenshot shows a software interface for security auditing. At the top, it displays 'Filter Set: Security Auditor View' and a count of 262 critical issues. Below this, a 'Group By:' dropdown menu is open, showing a list of categories such as '<none>', 'Category', 'Analysis Type', 'Analyzer', 'Analyzer-Category', 'Cat-Source', 'Category Analyzer', 'Correlated', 'Correlation Group', 'CWE', 'File Name', 'FISMA', 'Fortify Priority Order', 'Kingdom', 'Manual', 'New Issue', 'NIST SP 800-53 Rev.4', 'OWASP-2013->Category', 'OWASP Top 10 2004', 'OWASP Top 10 2007', 'OWASP Top 10 2010', 'OWASP Top 10 2013', 'Package', 'PCI 1.1', 'PCI 1.2', 'PCI 2.0', 'PCI 3.0', 'Priority by Category', 'RTA Protected', 'SANS Top 25 2009', 'SANS Top 25 2010', 'SANS Top 25 2011', 'Sink', 'Source', 'STIG-Category', 'STIG-Priority', 'STIG 3', 'STIG 3.4', 'STIG 3.5', 'STIG 3.6', 'STIG 3.7', 'Taint Flag', 'WASC 24 + 2', and 'Edit...'. The 'File Name' option is highlighted with a red arrow. To the right of the dropdown, there are several search filters like 'Password - [0 / 2]' and 'Field - [0 / 1]'. At the bottom, there is a search bar and a 'Manual' button.

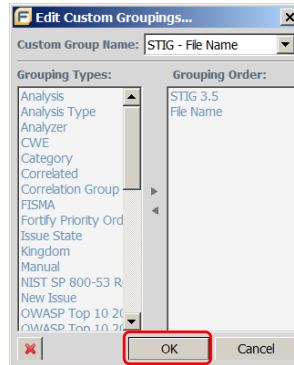
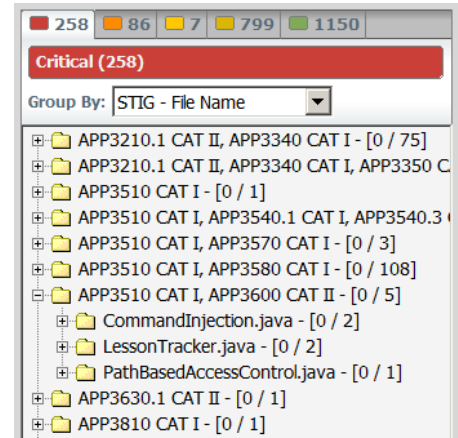
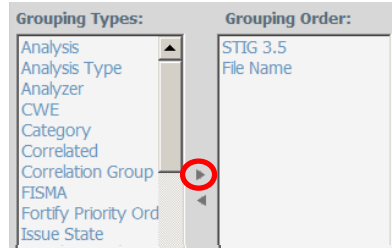
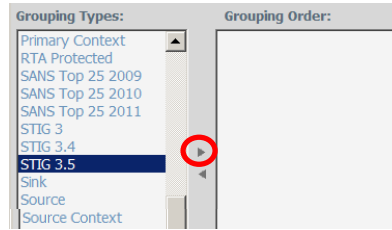
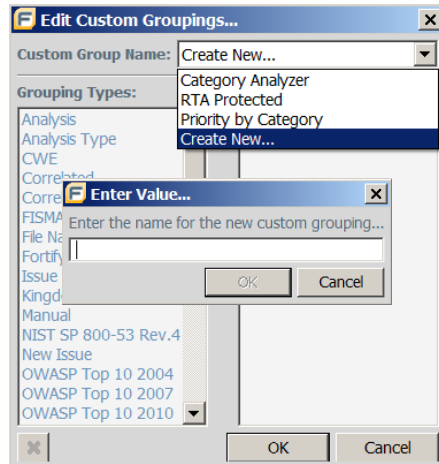
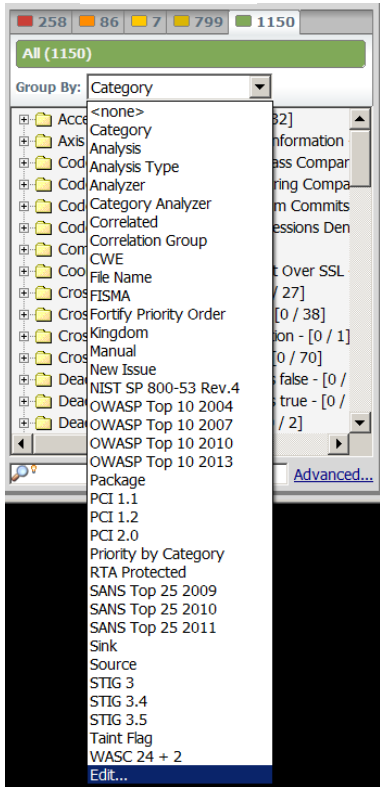
File Name

- AbstractLesson.java - [0 / 32]
- AccessControlMatrix.java - [0 / 2]
- BackDoors.java - [0 / 10]
- BasicAuthentication.java - [0 / 7]
- BlindSqlInjection.java - [0 / 8]
- BufferOverflow.java - [0 / 1]
- Category.java - [0 / 2]
- Challenge2Screen.java - [0 / 23]

DISA STIG

- APP3510 CAT I, APP3570 CAT I - [0 / 3]
- APP3510 CAT I, APP3580 CAT I - [0 / 108]
- APP3510 CAT I, APP3600 CAT II - [0 / 5]
- APP3630.1 CAT II - [0 / 1]
- APP3810 CAT I - [0 / 1]

Issue Grouping - custom



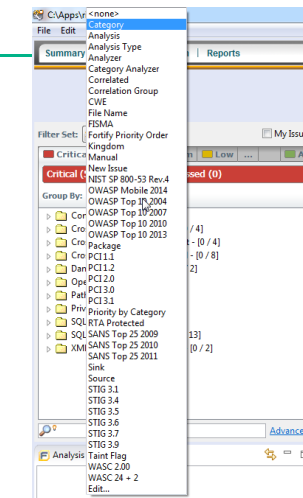
Exercise 8: Issue Grouping

1

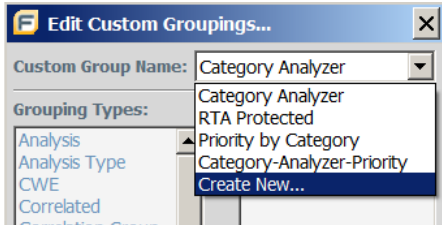
Create a 2-level grouping

FISMA

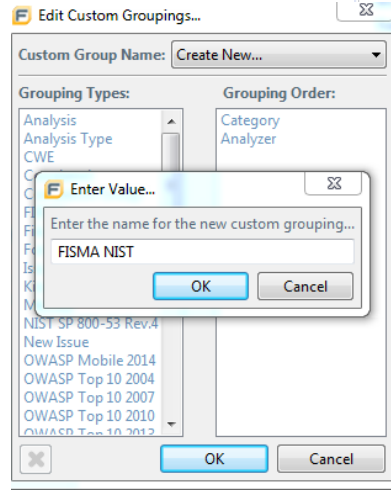
NIST 800-53



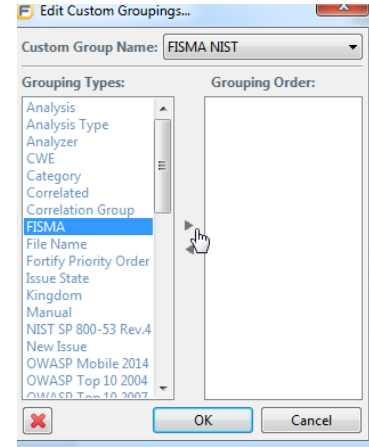
2



3



4



Working with the results . . .

The screenshot displays the Fortify Audit Workbench interface for a project named 'WebGoat5.0'. The main window is divided into several panes:

- Left Pane (Issues):** A tree view showing a total of 258 critical issues. The issues are grouped by category, with 'Cross-Site Scripting: Persistent' having 38 issues. Specific issues listed include 'BackDoors.java:125 (Cross-Site Scripting: Persistent)', 'CSRF.java:193 (Cross-Site Scripting: Persistent)', and 'DatabaseUtilities.java:154 (Shared Sink) - [0 / 11]'.
- Center Pane (Source Code):** A code editor showing a snippet of Java code from 'BackDoors.java'. The code includes a `catch` block for `Exception ex` and a call to `ec.addElement(new PRF(ex.getMessage()))`. A large text overlay 'Source Code' is positioned over this pane.
- Right Pane (Functions):** A list of functions and packages, including 'Top-level functions', 'java.io', 'java.lang', 'java.sql', and 'javax.servlet'. A large text overlay 'Functions' is positioned over this pane.
- Bottom Pane (Issue Auditing):** A detailed view of a specific issue: 'BackDoors.java:126 (Cross-Site Scripting: Persistent)'. It includes an 'Analysis Evidence' section with a tree view of code execution steps (e.g., 'executeQuery(return)', 'Assignment to rs', 'getString(this : return)'). A large text overlay 'Issue Auditing' is positioned over this pane. To the right of the evidence is a 'Cross-Site Scripting: Persistent (Input Validation and Representation, Data flow)' section with a description: 'The method concept1() in BackDoors.java sends unvalidated data to a web browser on line 126, which can result in the browser executing malicious code.' Below this is a 'More Information...' and 'Recommendations...' link.

At the bottom left, the 'Rule ID' is 9483F80E-4AED-4006-9CDD-82B1C13747EE and the 'Taint Flags' are DATABASE, XSS. The word 'Enterprise' is visible at the very bottom left corner.

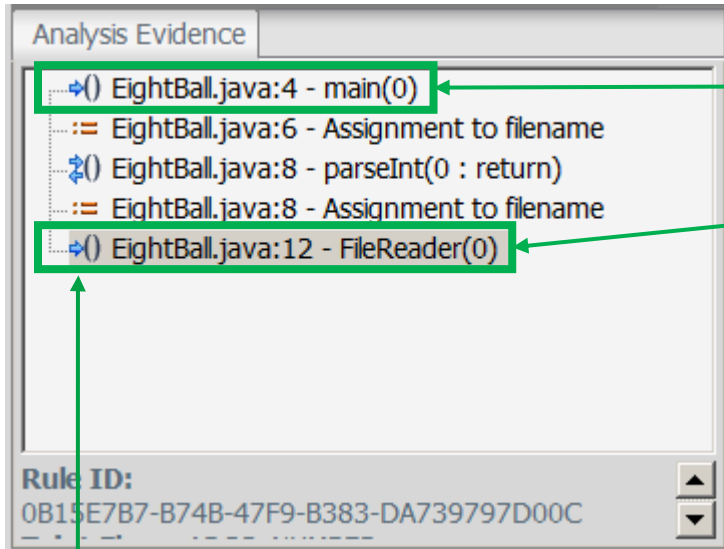
Working with the results . . .

The screenshot displays the Fortify Audit Workbench interface for a project named 'WebGoat5.0'. The main window is titled 'WebGoat5.0 - JavaSource/org/owasp/webgoat/lessons/BackDoors.java - Audit Workbench'. The interface is divided into several panes:

- Summary:** Shows a filter set of 'Security Auditor View' and a total of 258 critical issues. A tree view under 'Issues' lists various vulnerabilities, including 'Cross-Site Scripting: Persistent' and 'DatabaseUtilities.java:154 (Shared Sink)'. A large 'Issues' label is overlaid on this pane.
- Source Code:** Displays the source code for 'BackDoors.java'. Lines 122-136 show a loop of adding elements to a table. A 'catch' block is visible at lines 134-135. A large 'Source Code' label is overlaid on this pane.
- Issue Auditing:** The bottom pane shows details for an issue: 'BackDoors.java:126 (Cross-Site Scripting: Persistent)'. It includes a description: 'Cross-Site Scripting: Persistent (Input Validation and Representation, Data flow). The method concept1() in BackDoors.java sends unvalidated data to a web browser on line 126, which can result in the browser executing malicious code.' A large 'Issue Auditing' label is overlaid on this pane.
- Analysis Evidence:** The left pane shows a list of analysis evidence for the selected issue, including 'BackDoors.java:113 - executeQuery(return)', 'BackDoors.java:113 - Assignment to rs', 'BackDoors.java:126 - getString(this : return)', and 'BackDoors.java:126 - executeQuery(return)'. A large 'Analysis Evidence' label is overlaid on this pane.
- Functions:** The right pane shows a list of functions, including 'Top-level functions', 'java.io', 'java.lang', 'java.util', 'java.util.regex', 'javax.crypto', 'javax.servlet', and 'javax.servlet.jsp'. A large 'Functions' label is overlaid on this pane.

At the bottom left, the text 'Enterprise' is visible.











Analysis Evidence Panel












Trace of analysis evidence used in identifying the issue. From source to sink.

Icons are meaningful and documented in the “*About the Analysis Evidence Panel*” section of the **Audit Workbench User Guide**.

Analysis Evidence Panel

Icon	Description
	Data are assigned to a field or variable
	Information is read from a source external to the code (html form, url, and so on)
	Data are assigned to a globally scoped field or variable
	Comparison is made
	Function call receives tainted data
	Function call returns tainted data
	Passthrough, tainted data passes from one parameter to another in a function call
	An alias is created for a memory location
	Data are read from a variable
	Data are read from a global variable

Icon	Description
	Tainted data is returned from a function
	A pointer is created
	A pointer is dereferenced
	Scope of a variable ends
	Execution jumps
	Branch in the code's execution
	A branch is not taken in the code's execution
	Generic
	A runtime source, sink, or validation step

Issues Panel

The issue title is the last node in the analysis trace (sink)

The screenshot shows a software interface with two main panels. The top panel, titled "All (7)", displays a tree view of issues. The "Group By" dropdown is set to "Category". The tree structure is as follows:

- J2EE Bad Practices: Leftover Debug Code - [0 / 1]
- Path Manipulation - [0 / 2]
 - EightBall.java:12 (Shared Sink) - [0 / 2] (highlighted with a red box)
 - from EightBall.java:4 (Path Manipulation)
 - from EightBall.java:4 (Path Manipulation)
- Poor Logging Practice: Use of a System Output Str
- Unchecked Return Value - [0 / 1]
- Unreleased Resource: Streams - [0 / 1]

The bottom panel, titled "Analysis Evidence", shows a sequence of events in the analysis trace:

- EightBall.java:4 - main(0)
- EightBall.java:6 - Assignment to filename
- EightBall.java:8 - parseInt(0 : return)
- EightBall.java:8 - Assignment to filename
- EightBall.java:12 - FileReader(0) (highlighted with a red box)

At the bottom of the interface, the "Rule ID" is displayed as 0B15F7R7-R74R-47F9-R383-DA739797D00C.

Analysis Evidence Panel

Issues Panel

All (7)

Group By: Category

- J2EE Bad Practices: Leftover Debug Code - [0 / 1]
- Path Manipulation - [0 / 2]
 - EightBall.java:12 (Shared Sink) - [0 / 2]
 - from EightBall.java:4 (Path Manipulation)**
 - from EightBall.java:4 (Path Manipulation)
- Poor Logging Practice: Use of a System Output Str
- Unchecked Return Value - [0 / 1]
- Unreleased Resource: Streams - [0 / 1]

Advanced...

Analysis Evidence

- EightBall.java:4 - main(0)**
- EightBall.java:6 - Assignment to filename
- EightBall.java:8 - parseInt(0 : return)
- EightBall.java:8 - Assignment to filename
- EightBall.java:12 - FileReader(0)

Rule ID:
0B15F7R7-R74R-47F9-R383-DA739797D00C

Sub-group title is the first line of the Analysis Evidence

Analysis Evidence Panel

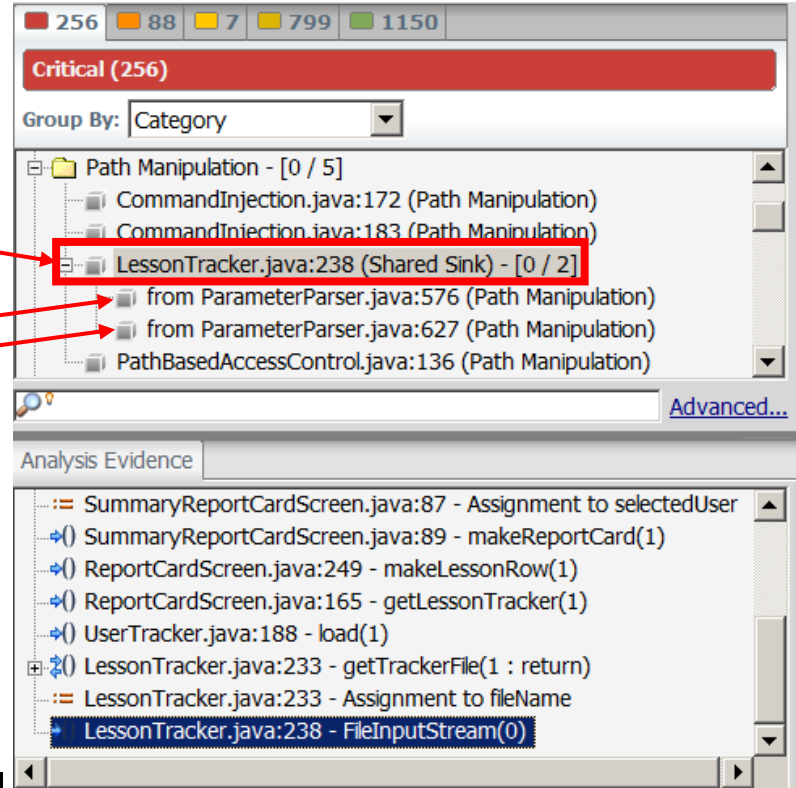
Issues Panel

Shared sink

Different sources

Two issues

Analysis Evidence Panel



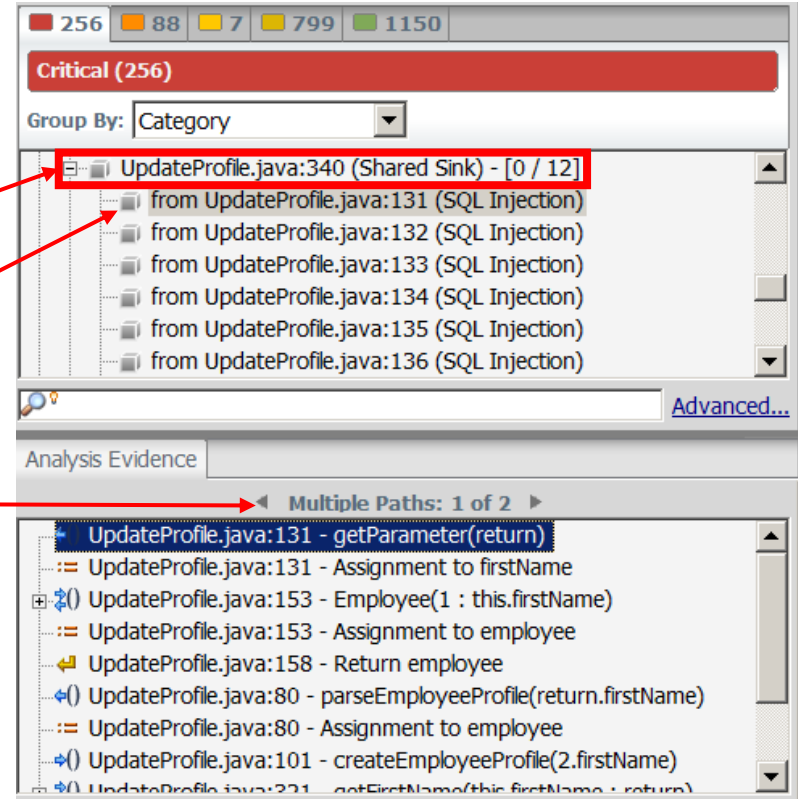
Issues Panel

One sink

One source

Multiple paths

One issue



Analysis Evidence Panel

Working with the results . . .

The screenshot displays the Fortify Audit Workbench interface. At the top, the title bar reads "WebGoat5.0 - JavaSource/org/owasp/webgoat/lessons/BackDoors.java - Audit Workbench". The main window is divided into several panes:

- Summary:** Shows a filter set of "Security Auditor View" and a total of 258 critical issues. A tree view under "Issues" lists various vulnerabilities, including "Cross-Site Scripting: Persistent" and "DatabaseUtilities.java:154 (Shared Sink)".
- Source Code:** Displays the Java source code for "BackDoors.java". Lines 122-136 are visible, showing the construction of an HTML table row with fields for "userid", "password", "ssn", and "salary".
- Issue Auditing:** The "Issue: BackDoors.java:126 (Cross-Site Scripting: Persistent)" pane is active. It shows the analysis path: "BackDoors.java:113 - executeQuery(return)" and "BackDoors.java:126 - getString(this : return)". The "Analysis Evidence" pane on the left shows the execution flow. The "Issue" pane provides a detailed description: "Cross-Site Scripting: Persistent (Input Validation and Representation, Data flow). The method concept1() in BackDoors.java sends unvalidated data to a web browser on line 126, which can result in the browser executing malicious code." It also includes a "More Information..." link.
- Functions:** A list of functions is shown on the right, including "Top-level functions", "java.io", "java.lang", "java.util", "java.util.regex", "javax.crypto", "javax.crypto.spec", "javax.servlet", "javax.servlet.http", and "javax.servlet.jsp".

Enterprise

Working with the results . . .

The screenshot displays the Fortify Audit Workbench interface for a project named 'WebGoat5.0'. The main window is divided into several panes:

- Summary:** Shows a total of 258 critical issues, with a breakdown of 90 high, 7 medium, 799 low, and 1154 informational issues. The 'Group By' is set to 'Category', showing a list of 'Cross-Site Scripting: Persistent' issues.
- Source Code:** Displays the source code for 'BackDoors.java', with lines 122-136 visible. The code shows a loop adding elements to a table, with a 'catch' block for exceptions.
- Issue Auditing:** A detailed view of an issue at line 126, categorized as 'Cross-Site Scripting: Persistent (Input Validation and Representation, Data flow)'. The analysis states that unvalidated data is sent to a web browser, which could result in malicious code execution. It includes a 'More Information...' and 'Recommendations...' link.
- Analysis Evidence:** A list of evidence items for the selected issue, including 'executeQuery(return)', 'Assignment to rs', and 'getString(this : return)'. The rule ID is 9483FB0E-4AED-4006-9CDD-82B1C13747EE and the taint flags are DATABASE, XSS.
- Functions:** A list of functions used in the code, including 'Top-level functions', 'java.io', 'java.lang', 'java.sql', 'java.text', 'java.util', 'java.util.regex', 'javax.crypto', 'javax.crypto.spec', 'javax.servlet', 'javax.servlet.http', and 'javax.servlet.jsp'.

Issue Auditing Panel - Summary

The screenshot shows the 'Summary' tab of an issue auditing panel. The interface includes a navigation bar with tabs for Summary, Details, Recommendations, History, Diagram, and Filters. The main content area displays the issue title 'AbstractLesson.java:920 (Cross-Site Scripting: Reflected)'. On the left, there is a 'User' field and an 'Analysis' dropdown menu with options: 'Not an Issue', 'Reliability Issue', 'Bad Practice', 'Suspicious', and 'Exploitable'. Below the analysis menu are two buttons: a red 'X' icon and a bug icon. The central area contains a large text input field for comments, with a 'Click to append comment (Ctrl+Enter to save)' label and a 'Submit' button. On the right, a detailed description of the issue is shown, including the title 'Cross-Site Scripting: Reflected (Input Validation and Representation, Data flow)' and a paragraph of text explaining the vulnerability. Below the description are links for 'More Information...' and 'Recommendations...'. Annotations with red arrows and boxes point to various elements: 'Set the analysis tag' points to the Analysis dropdown; 'List of saved comments' points to the central text input field; 'Short description of the selected issue' points to the right-hand description box; 'Suppress the issue' points to the red 'X' button; 'Submit to bug tracking system here' points to the bug icon; 'Enter comments here' points to the central text input field; and 'Commit or cancel current comment' points to the 'Submit' button.

Set the analysis tag

List of saved comments

Short description of the selected issue

Suppress the issue

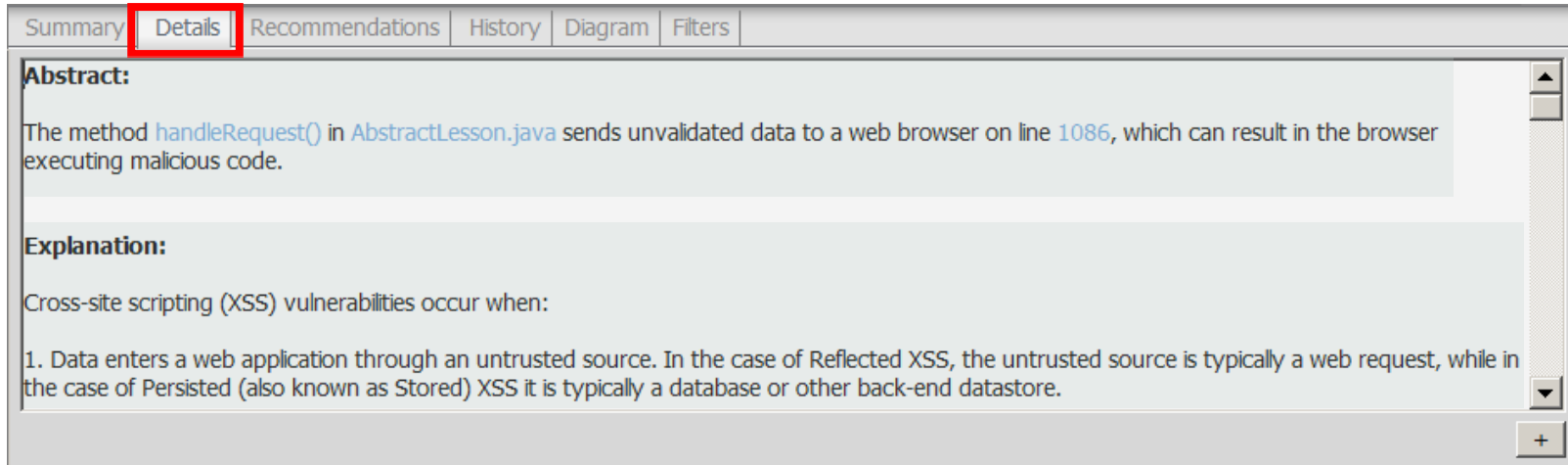
Submit to bug tracking system here

Enter comments here

Commit or cancel current comment

Issue Auditing Panel - Details

Code specific abstract of selected issue.



Summary **Details** Recommendations History Diagram Filters

Abstract:

The method `handleRequest()` in `AbstractLesson.java` sends unvalidated data to a web browser on line 1086, which can result in the browser executing malicious code.

Explanation:

Cross-site scripting (XSS) vulnerabilities occur when:

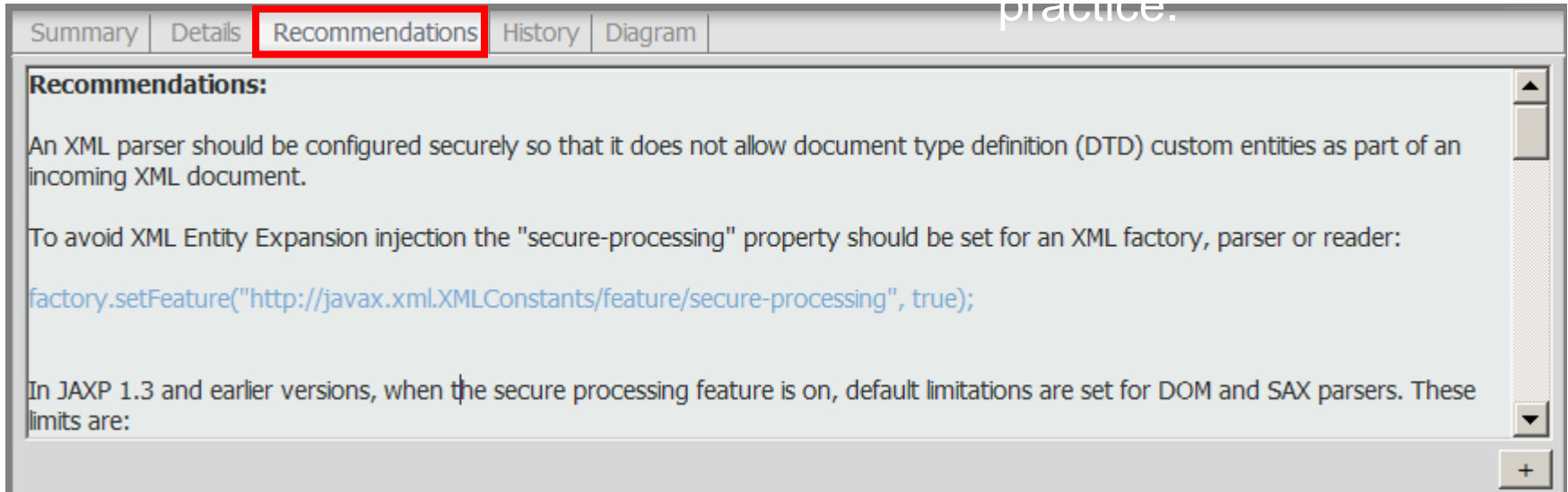
1. Data enters a web application through an untrusted source. In the case of Reflected XSS, the untrusted source is typically a web request, while in the case of Persisted (also known as Stored) XSS it is typically a database or other back-end datastore.

Detailed description of the selected issue.

Issue Auditing Panel - Recommendations

Suggestions and examples of how to secure the vulnerability or remedy the bad

practice.



The screenshot shows a software interface with a tabbed menu at the top. The tabs are 'Summary', 'Details', 'Recommendations', 'History', and 'Diagram'. The 'Recommendations' tab is selected and highlighted with a red border. Below the tabs, the main content area is titled 'Recommendations:' and contains the following text:

An XML parser should be configured securely so that it does not allow document type definition (DTD) custom entities as part of an incoming XML document.

To avoid XML Entity Expansion injection the "secure-processing" property should be set for an XML factory, parser or reader:

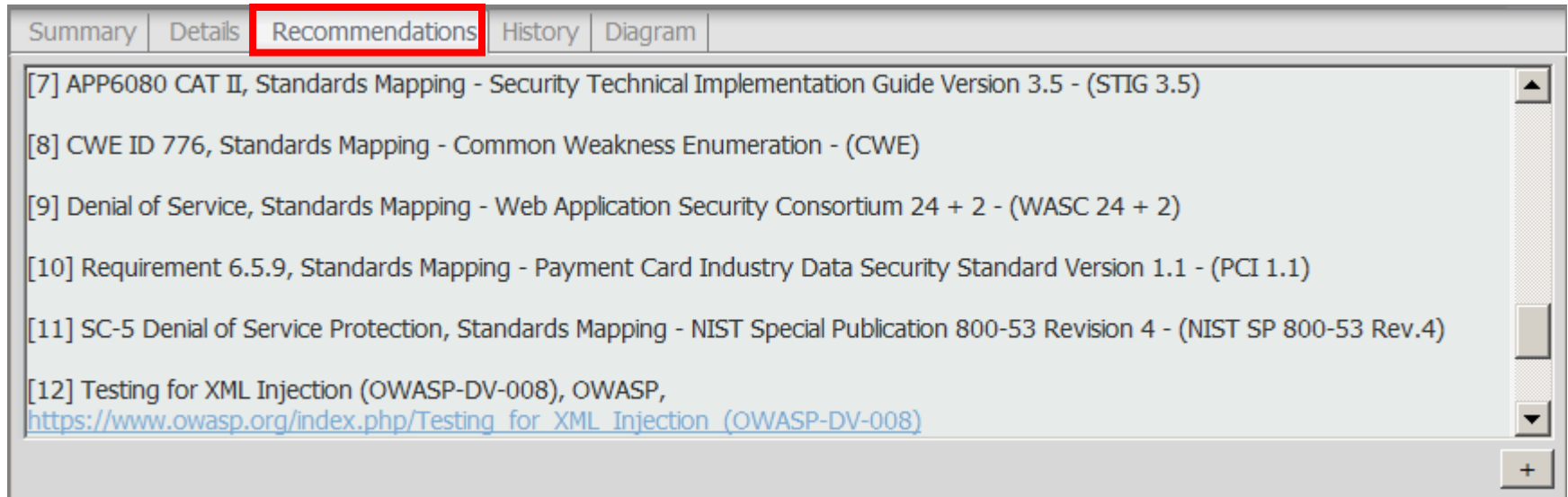
```
factory.setFeature("http://javax.xml.XMLConstants/feature/secure-processing", true);
```

In JAXP 1.3 and earlier versions, when the secure processing feature is on, default limitations are set for DOM and SAX parsers. These limits are:

On the right side of the content area, there are vertical scroll bars with up and down arrows, and a '+' button at the bottom right corner.

Issue Auditing Panel - Recommendations

Scroll down to see mappings and external research references.



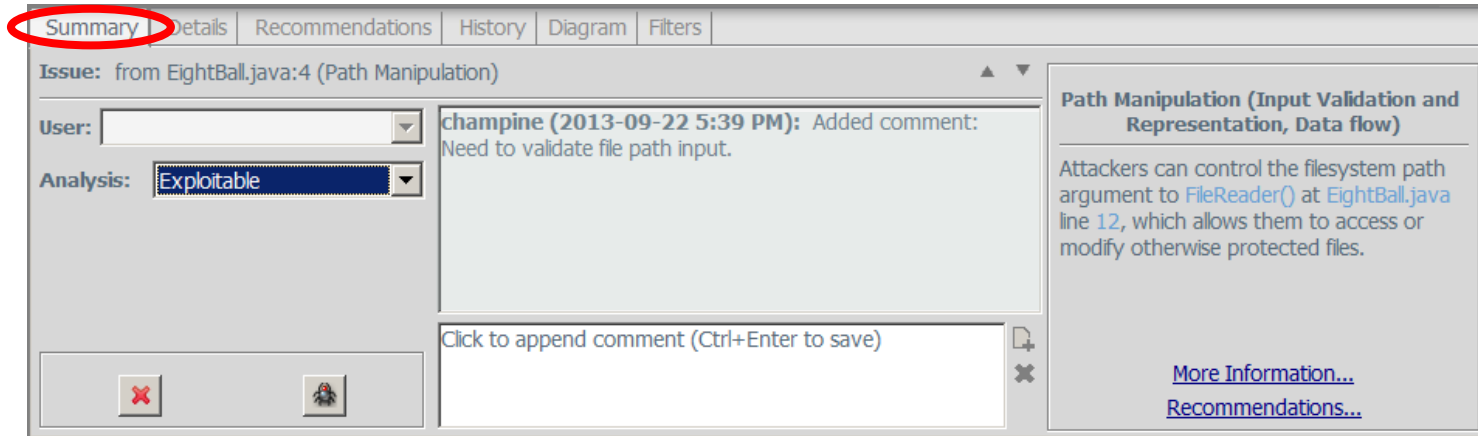
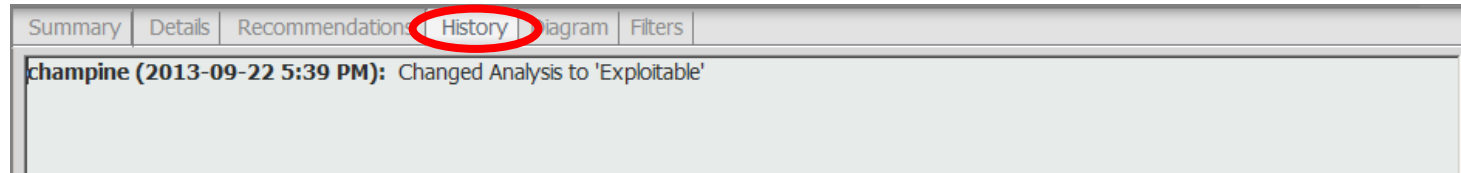
The screenshot shows a web interface with a tabbed menu at the top. The 'Recommendations' tab is selected and highlighted with a red border. Below the tabs is a scrollable list of references. The list contains six items, each starting with a bracketed number in the range [7] to [12]. The last item includes a blue hyperlink to the OWASP website. On the right side of the list, there are standard scroll controls: an upward arrow, a vertical scrollbar, a downward arrow, and a plus sign button at the bottom.

Summary | Details | **Recommendations** | History | Diagram

- [7] APP6080 CAT II, Standards Mapping - Security Technical Implementation Guide Version 3.5 - (STIG 3.5)
- [8] CWE ID 776, Standards Mapping - Common Weakness Enumeration - (CWE)
- [9] Denial of Service, Standards Mapping - Web Application Security Consortium 24 + 2 - (WASC 24 + 2)
- [10] Requirement 6.5.9, Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 - (PCI 1.1)
- [11] SC-5 Denial of Service Protection, Standards Mapping - NIST Special Publication 800-53 Revision 4 - (NIST SP 800-53 Rev.4)
- [12] Testing for XML Injection (OWASP-DV-008), OWASP, [https://www.owasp.org/index.php/Testing for XML Injection \(OWASP-DV-008\)](https://www.owasp.org/index.php/Testing_for_XML_Injection_(OWASP-DV-008))

Issue Auditing Panel - History

Activities such as changing the analysis tag and suppressing an issue are logged.

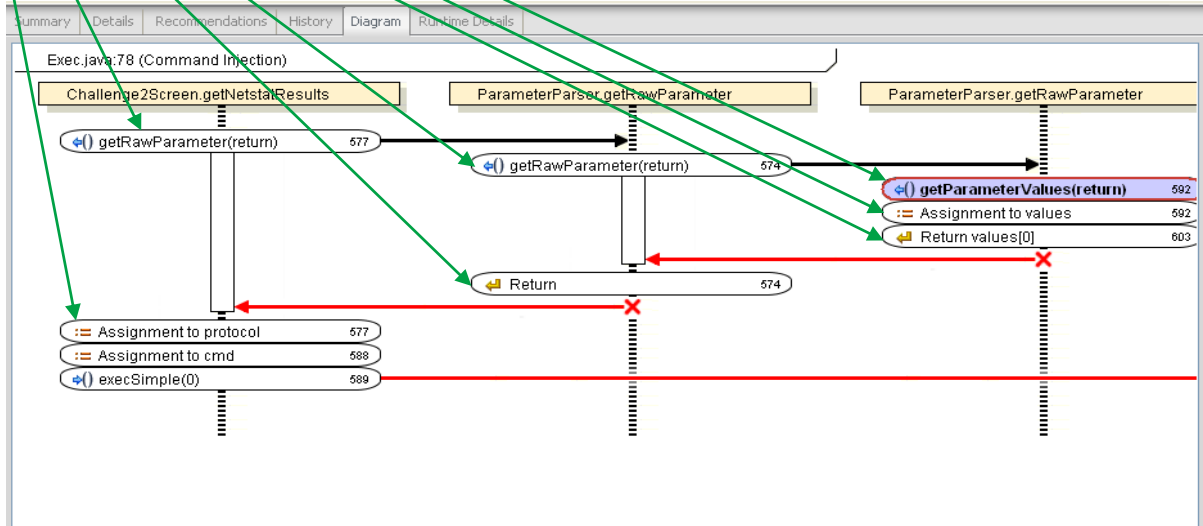


Issue Auditing Panel - Diagram

- ParameterParser.java:592 - getParameterValues(return)
- ParameterParser.java:592 - Assignment to values
- ParameterParser.java:603 - Return values[0]
- ParameterParser.java:574 - getRawParameter(return)
- ParameterParser.java:574 - Return
- Challenge25screen.java:577 - getRawParameter(return)
- Challenge25screen.java:577 - Assignment to protocol

Red lines in the diagram are the flow of tainted data

Standard UML call graph of the Analysis



Working with the results . . .

The screenshot displays the Fortify Audit Workbench interface for a project named 'WebGoat5.0'. The main window is divided into several panes:

- Summary:** Shows a filter set of 'Security Auditor View' and a total of 258 critical issues. A tree view on the left lists issues by category, such as 'Cross-Site Scripting: Persistent' and 'DatabaseUtilities.java:154 (Shared Sink)'. A large 'Issues' label is overlaid on this pane.
- Source Code:** The central pane shows the source code for 'BackDoors.java'. Lines 122-136 are visible, showing a loop that adds elements to a table. A large 'Source Code' label is overlaid on this pane.
- Issue Auditing:** The bottom pane shows details for an issue: 'BackDoors.java:126 (Cross-Site Scripting: Persistent)'. It includes a description of the issue, analysis evidence, and a recommendation to append a comment. A large 'Issue Auditing' label is overlaid on this pane.
- Functions:** The right-hand pane shows a list of functions, including 'Top-level functions', 'java.io', 'java.lang', and 'java.util'. A large 'Functions' label is overlaid on this pane.

At the bottom left, the 'Analysis Evidence' pane shows a list of evidence items, such as 'BackDoors.java:113 - executeQuery(return)'. A large 'Analysis Evidence' label is overlaid on this pane.

At the bottom left, the 'Rule ID: 9483FB0E-4AED-4006-9CDD-82B1C13747EE' and 'Taint Flags: DATABASE, XSS' are displayed. The 'Enterprise' logo is visible in the bottom left corner.

Working with the results . . .

The screenshot displays the Fortify Audit Workbench interface for a project named 'WebGoat5.0'. The main window is titled 'WebGoat5.0 - JavaSource/org/owasp/webgoat/lessons/BackDoors.java - Audit Workbench'. The interface is divided into several panes:

- Summary:** Shows a filter set of 'Security Auditor View' and a total of 258 critical issues. A tree view under 'Issues' lists various findings, such as 'Cross-Site Scripting: Persistent' and 'DatabaseUtilities.java:154 (Shared Sink)'. A large 'Issues' label is overlaid on this pane.
- Source Code:** Displays the source code for 'BackDoors.java'. Lines 122-136 are visible, showing HTML element creation and a catch block. A large 'Source Code' label is overlaid on this pane.
- Issue Auditing:** The 'Issue: BackDoors.java:126 (Cross-Site Scripting: Persistent)' pane is active. It shows the user, analysis, and a detailed description of the issue: 'Cross-Site Scripting: Persistent (Input Validation and Representation, Data flow). The method concept1() in BackDoors.java sends unvalidated data to a web browser on line 126, which can result in the browser executing malicious code.' A large 'Issue Auditing' label is overlaid on this pane.
- Analysis Evidence:** The 'Analysis Evidence' pane shows a list of code snippets related to the issue, such as 'BackDoors.java:113 - executeQuery(return)' and 'BackDoors.java:126 - getString(this : return)'. A large 'Analysis Evidence' label is overlaid on this pane.
- Functions:** The 'Functions' pane on the right lists various functions and packages, including 'Top-level functions', 'java.io', 'java.lang', and 'java.util'. A large 'Functions' label is overlaid on this pane.

At the bottom left, the 'Rule ID: 9483F80E-4AED-4006-9CDD-82B1C13747EE' and 'Taint Flags: DATABASE, XSS' are visible. The 'Enterprise' logo is in the bottom left corner.

Source Code Panel

Clicking an issue syncs the source code panel to the file and line number of the sink.

The screenshot displays a security tool interface. On the left, a 'Filter Set' is set to 'Security Auditor View'. Below this, there are colored bars representing issue counts: 258 (Critical), 86 (High), 7 (Medium), 799 (Low), and 1150 (Info). The 'Group By' is set to 'Category'. A tree view shows a folder 'Cross-Site Scripting: Reflected - [0 / 70]' containing several items, with 'AbstractLesson.java:1086 (Cross-Site Sc)' selected. A red arrow points from this item to the source code panel on the right. The source code panel shows the file 'AbstractLesson.java' with line numbers 1072 to 1091. The code is as follows:

```
1072
1073
1074     /**
1075      * Description of the Method
1076      *
1077      * @param s
1078      *         Description of the Parameter
1079      */
1080     public void handleRequest(WebSession s)
1081     {
1082         // call createContent first so messages will go somewhere
1083
1084         Form form = new Form(getFormAction(), Form.POST).setName("form")
1085             .setEncType("");
1086         form.addElement(createContent(s));
1087
1088         setContent(form);
1089     }
1090
1091
```

Source Code Panel

The screenshot displays a security tool interface with several panels:

- Filter Set:** Security Auditor View, My Issues (unchecked).
- Issue Counts:** 258 Critical, 86, 7, 799, 1150.
- Group By:** Category.
- Issue Tree:** Cross-Site Scripting: Reflected - [0 / 70].
 - AbstractLesson.java:872 (Shared Sink)
 - AbstractLesson.java:920 (Cross-Site Scri)
 - AbstractLesson.java:1086 (Cross-Site Sc
 - BackDoors.java:235 (Cross-Site Scripting)
 - BasicAuthentication.java:143 (Cross-Site
 - BasicAuthentication.java:145 (Cross-Site
 - BlindSqlInjection.java:83 (Cross-Site Scri
- Analysis Evidence:** Multiple Paths: 1 of 3.
 - ParameterParser.java:608 - getRowParamet
 - ParameterParser.java:608 - Return
 - ThreadSafetyProblem.java:83 - getRowPara
 - ThreadSafetyProblem.java:83 - Assignment
 - ThreadSafetyProblem.java:84 - Assignment** (highlighted)
 - ThreadSafetyProblem.java:107 - addElemen
 - ThreadSafetyProblem.java:131 - Return ec
- Source Code Panel:** AbstractLesson.java. Line 84 is highlighted, corresponding to the selected issue in the Analysis Evidence panel. The code snippet is:

```
84 originalUser = currentUser;
```
- Issue Details Panel:** Issue: AbstractLesson.java:1086 (Cross-Site Scripting: Reflector). Fields for User and Analysis are present.
- Right Panel:** Cross-Site Scripting: Reflected (Input Validation and Representation, Data flow). Includes links for More Information... and Recommendations...

Clicking on a trace node syncs the source code panel to node's file and line number.

Source Code Panel - Project Summary

WebGoat5.0 - C:\Users\champine\AppData\Local\Fortify\AWB-4.00\WebGoat5.0\WebGoat5.0.fpr - Audit Workbench

File Edit Tools Options Help

Project Summary

Audit Guide...
Generate Report...
Calculate Hotspot Ranking
Merge Audit Projects...
Configure Upload...
Upload Audit Project
Configure Source Path...
Extract Source Code...
Select Bugtracker...
Disconnect Bugtracker...
Project Configuration...

AUDIT WORKBENCH FORTIFY

Summary Certification Runtime Analysis Build Information Analysis Information

Build ID: WebGoat5.0 **Scanned:** 188 files, 9,564 LOC (Executable)
Scan Date: Sep 16, 2013 **Total Issues:** 1,150
Warnings: None **Certification:** Results Certification Valid

All issues by Folder

Severity	Count
Medium	7
High	88
Critical	256
Low	799

Summary Details Recommendations History Diagram Filters

Issue: AbstractLesson.java:920 (Cross-Site Scripting: Reflected)

User: [Dropdown]

Cross-Site Scripting: Reflected (Input Validation and Representation, Data flow)

[More Information...](#)
[Recommendations...](#)

Click to append comment (Ctrl+Enter to save)

Rule ID: DBDDBFC6-DE26-4FC5-8347-D48032B2BF5D
Taint Flags: NO_NEW_LINE, WEB, XSS

Source Code Panel - Project Summary

Project Summary

Summary | Certification | Runtime Analysis | Build Information | Analysis Information

Build ID: WebGoat5.0
Scan Date: Sep 16, 2013
Warnings: None

Scanned: 188 files, 9,564 LOC (Executable)
Total Issues: 1,150
Certification: Results Certification Valid

All issues by Folder

Severity	Count
Low	799
Critical	256
High	88
Medium	7

If someone tries to tamper with the FPR file directly, result certification will become invalid

Source Code Panel - Project Summary

Project Summary

Summary | Certification | Runtime Analysis | **Build Information** | Analysis Information

Build ID: WebGoat5.0 **Build Label:** <No Build Label>
Files: 188 **Source Last Modified Date:** Aug 9, 2013
Executable LOC: 9,564 **Total LOC:** 32,613

File Name	LOC	File Size	Date
JavaSource/org/owasp/webgoat/HammerHead.java	124 Lines	15.1 KB	Aug 9, 2013 4:03:54 PM
JavaSource/org/owasp/webgoat/LessonSource.java	50 Lines	5.7 KB	Aug 9, 2013 4:03:54 PM
JavaSource/org/owasp/webgoat/lessons/AbstractLesson.java	199 Lines	27.1 KB	Aug 9, 2013 4:03:54 PM
JavaSource/org/owasp/webgoat/lessons/AccessControlMatrix.java	66 Lines	7.4 KB	Aug 9, 2013 4:03:54 PM
JavaSource/org/owasp/webgoat/lessons/BackDoors.java	111 Lines	8.8 KB	Aug 9, 2013 4:03:54 PM
JavaSource/org/owasp/webgoat/lessons/BasicAuthentication.java	104 Lines	10.3 KB	Aug 9, 2013 4:03:54 PM

Classpath

C:\Program Files\HP_Fortify\HP_Fortify_SCA_and_Apps_4.0.0\bin\sourceanalyzer.jar
C:\Program Files\HP_Fortify\HP_Fortify_SCA_and_Apps_4.0.0\bin\sourceanalyzer.jar
C:\Program Files\HP_Fortify\HP_Fortify_SCA_and_Apps_4.0.0\bin\sourceanalyzer.jar
C:\Program Files\HP_Fortify\HP_Fortify_SCA_and_Apps_4.0.0\bin\sourceanalyzer.jar
C:\Program Files\HP_Fortify\HP_Fortify_SCA_and_Apps_4.0.0\bin\sourceanalyzer.jar

Executable LOC: SCA doesn't count blank lines and comments
Total LOC: SCA doesn't include HTML, XML and properties files

The list of all scanned files. Same as # sourceanalyzer -b build_id -show-

Working with the results . . .

The screenshot displays the Fortify Audit Workbench interface for a Java project. The main window is titled "WebGoat5.0 - JavaSource/org/owasp/webgoat/lessons/BackDoors.java - Audit Workbench". The interface is divided into several panes:

- Summary:** Shows a filter set of "Security Auditor View" and a total of 258 critical issues.
- Issues:** A tree view showing a list of issues, primarily categorized as "Cross-Site Scripting: Persistent".
- Source Code:** A code editor showing the source code for "BackDoors.java", with lines 126-127 highlighted. The code includes a loop that adds elements to a table, with a catch block for exceptions.
- Issue Auditing:** A detailed view of a specific issue: "BackDoors.java:126 (Cross-Site Scripting: Persistent)". It includes a description of the issue, a "User:" field, and an "Analysis:" field. A tooltip explains that the method `concept1()` sends unvalidated data to a web browser, which can result in malicious code execution.
- Functions:** A list of functions used in the code, including `java.io`, `java.lang`, `java.sql`, and `javax.servlet`.

At the bottom left, the "Rule ID" is 9483F80E-4AED-4006-9CDD-82B1C13747EE and the "Taint Flags" are DATABASE, XSS.

Working with the results . . .

The screenshot displays the Fortify Audit Workbench interface for a project named 'WebGoat5.0'. The main window is divided into several panes:

- Summary:** Shows a filter set of 'Security Auditor View' and a total of 258 critical issues. A tree view under 'Issues' lists various vulnerabilities, including 'Cross-Site Scripting: Persistent' and 'DatabaseUtilities.java:154 (Shared Sink)'.
- Source Code:** Displays the source code for 'BackDoors.java', with lines 122-136 highlighted. The code shows a loop adding elements to a table, with a 'catch' block for exceptions.
- Issue Auditing:** Shows a detailed view of an issue: 'BackDoors.java:126 (Cross-Site Scripting: Persistent)'. It includes a description of the issue, a 'User' field, and an 'Analysis' field. A note explains that the method 'concept1()' sends unvalidated data to a web browser on line 126, which can result in malicious code execution.
- Analysis Evidence:** Lists evidence for the selected issue, including 'BackDoors.java:113 - executeQuery(return)', 'BackDoors.java:113 - Assignment to rs', 'BackDoors.java:126 - getString(this : return)', and 'BackDoors.java:126 - return'.
- Functions:** A sidebar on the right lists various functions and packages, including 'Top-level functions', 'java.io', 'java.lang', 'java.net', 'java.nio', 'java.security', 'java.sql', 'java.text', 'java.util', 'java.util.regex', 'javax.crypto', 'javax.crypto.spec', 'javax.servlet', 'javax.servlet.http', and 'javax.servlet.jsp'.

At the bottom left, the 'Enterprise' logo is visible.

Functions Panel

Lists all functions that were declared or called in the source code scanned.

Package

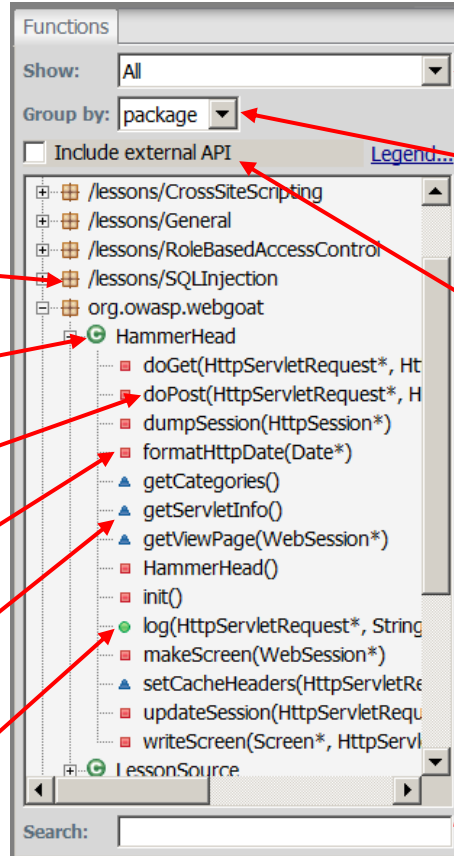
Class

Function Name

Not Covered by

Covered by Rules

Covered by and Matching Rules



Show all functions or only those not covered by rules.

Group by package, class or just list functions.

Include external API's used in the source code scanned.

Search packages, classes and functions

Working with the results . . .

The screenshot displays the Fortify Audit Workbench interface for a project named 'WebGoat5.0'. The main window is divided into several panes:

- Left Pane (Issues):** Shows a summary of 258 critical issues. A tree view lists specific issues, such as 'Cross-Site Scripting: Persistent' in 'BackDoors.java:125' and 'DatabaseUtilities.java:154 (Shared Sink)'. A filter set is set to 'Security Auditor View'.
- Center Pane (Source Code):** Displays the source code for 'BackDoors.java'. The code shows a loop adding elements to a table, with a 'catch' block for exceptions. A large text overlay 'Source Code' is present.
- Right Pane (Functions):** Lists various functions and packages, including 'Top-level functions', 'java.io', 'java.lang', and 'java.util'. A large text overlay 'Functions' is present.
- Bottom Pane (Issue Auditing):** Shows details for a specific issue: 'Issue: BackDoors.java:126 (Cross-Site Scripting: Persistent)'. It includes a description of the issue, a 'User' field, an 'Analysis' field, and a 'More Information...' link. A large text overlay 'Issue Auditing' is present.

At the bottom left, the 'Analysis Evidence' pane shows a list of code snippets related to the issue, such as 'BackDoors.java:113 - executeQuery(return)' and 'BackDoors.java:126 - getString(this : return)'. A large text overlay 'Analysis Evidence' is present.



Other Features

Suppression

A way to hide an issue from view and consider it audited.

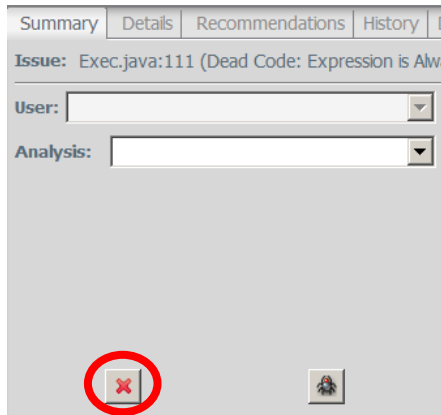
What might be suppressed?

- Issues that you are certain will not be of concern.
- Issues that you plan to never fix.
- Issues that are warnings concerning code quality or correctness (lower priority).

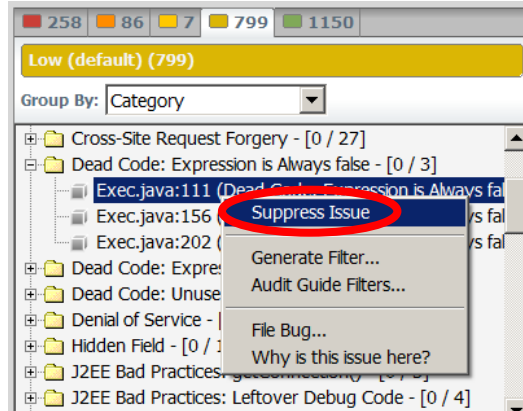
Suppression

Three ways to suppress

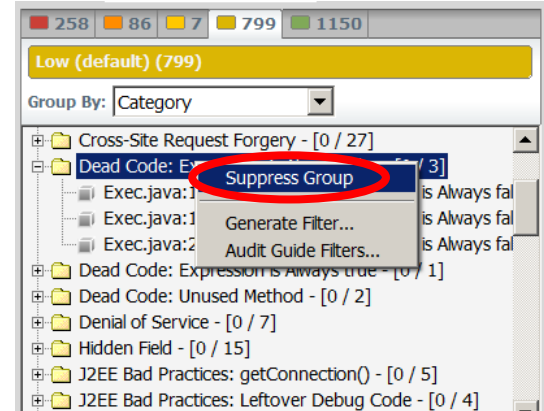
Suppress the currently



Right click on an issue

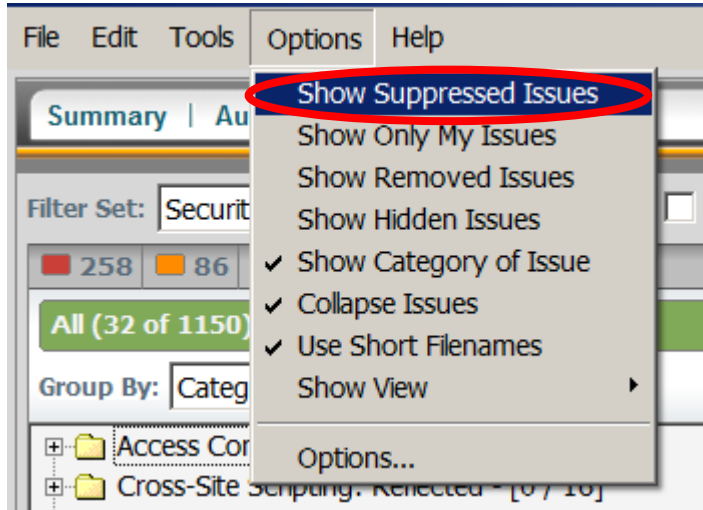


Right click on a group of issues



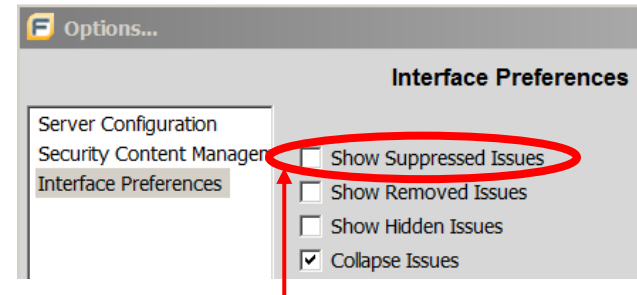
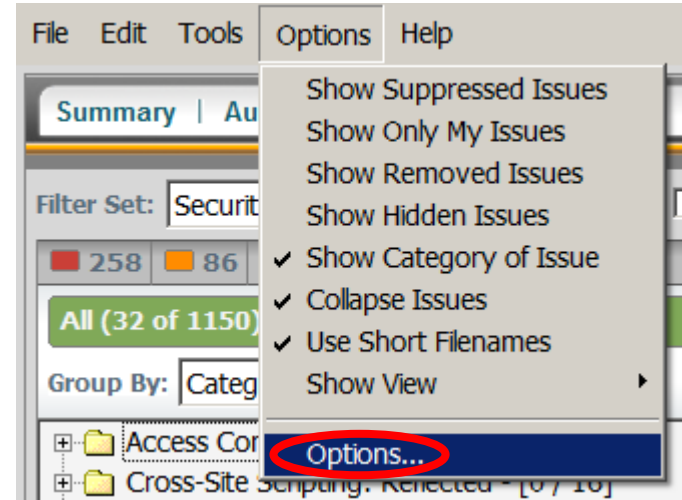
Suppression

How to view suppressed issues



Toggles showing suppressed issues

OR

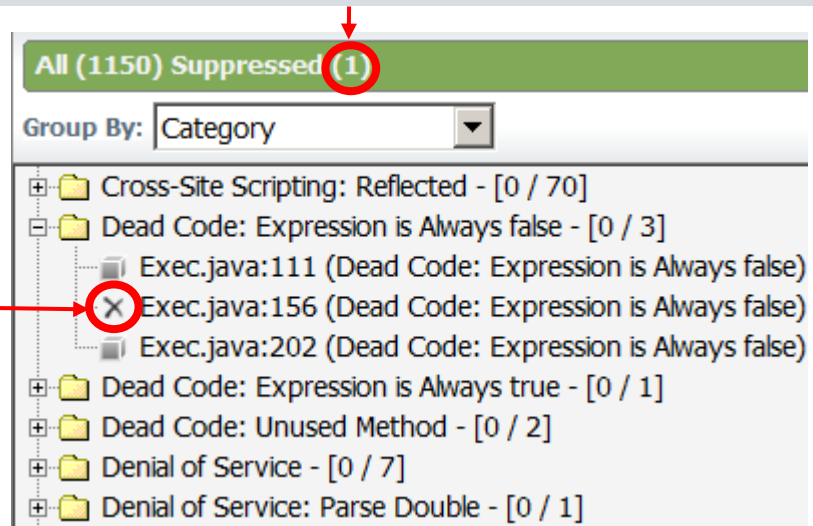


Interface will display suppressed issues by default when set.

Suppression

Viewing suppressed issues

Total suppressed issues in current folder

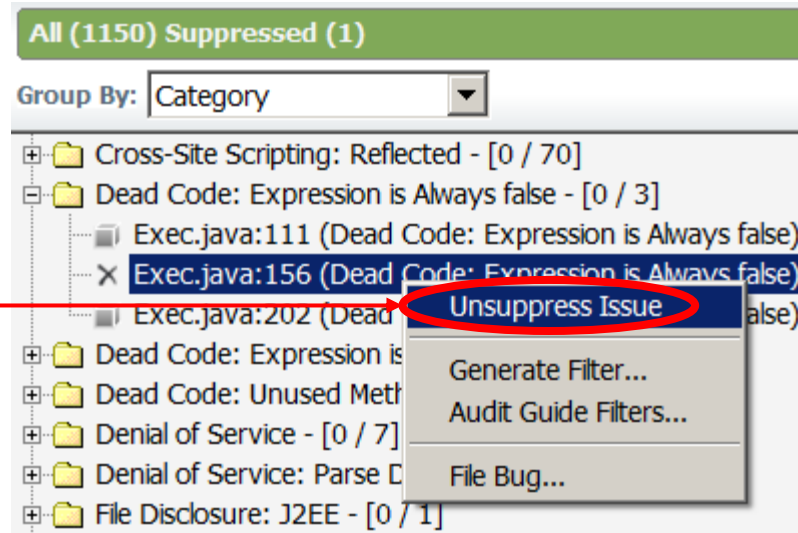


The icon for suppressed issue

Suppression

Unsuppress an issue

While viewing suppressed issues, right click on a suppressed issue.



Exercise 9: Audit and Suppress

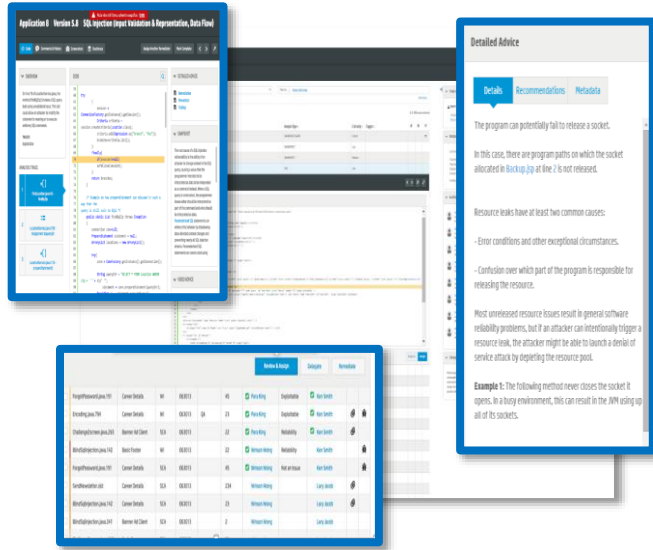
- Add a comment to all issues related to:
- Suppress all Dead Code



Software Security Center Overview

HPE Fortify Software Security Center

Management, tracking and remediation of enterprise software risk



Features:

- Specify, communicate and track security activities performed on projects
- Role-based, process-driven management of software security program
- Flexible repository and exporting platform for security status, trending and compliance

Benefits:

- Provides a clear, accurate picture of software risk across the enterprise
- Lowers cost of resolving vulnerabilities
- Identify areas of improvement for accelerated reduction of risk and costs

Problem it solves:

Provides visibility into security activities within development

HPE Security Fortify Software Security Center

Vulnerability detail

The screenshot displays the HPE Security Fortify Software Security Center interface. The main window shows a vulnerability detail for 'Application 8 Version 5.8 SQL Injection (Input Validation & Representation, Data Flow)'. The code editor highlights a line of code: `socket = SocketFactory.getDefault().getSocket();`. The detailed advice panel provides context on resource leaks and socket management. Below the code editor, a table lists identified vulnerabilities.

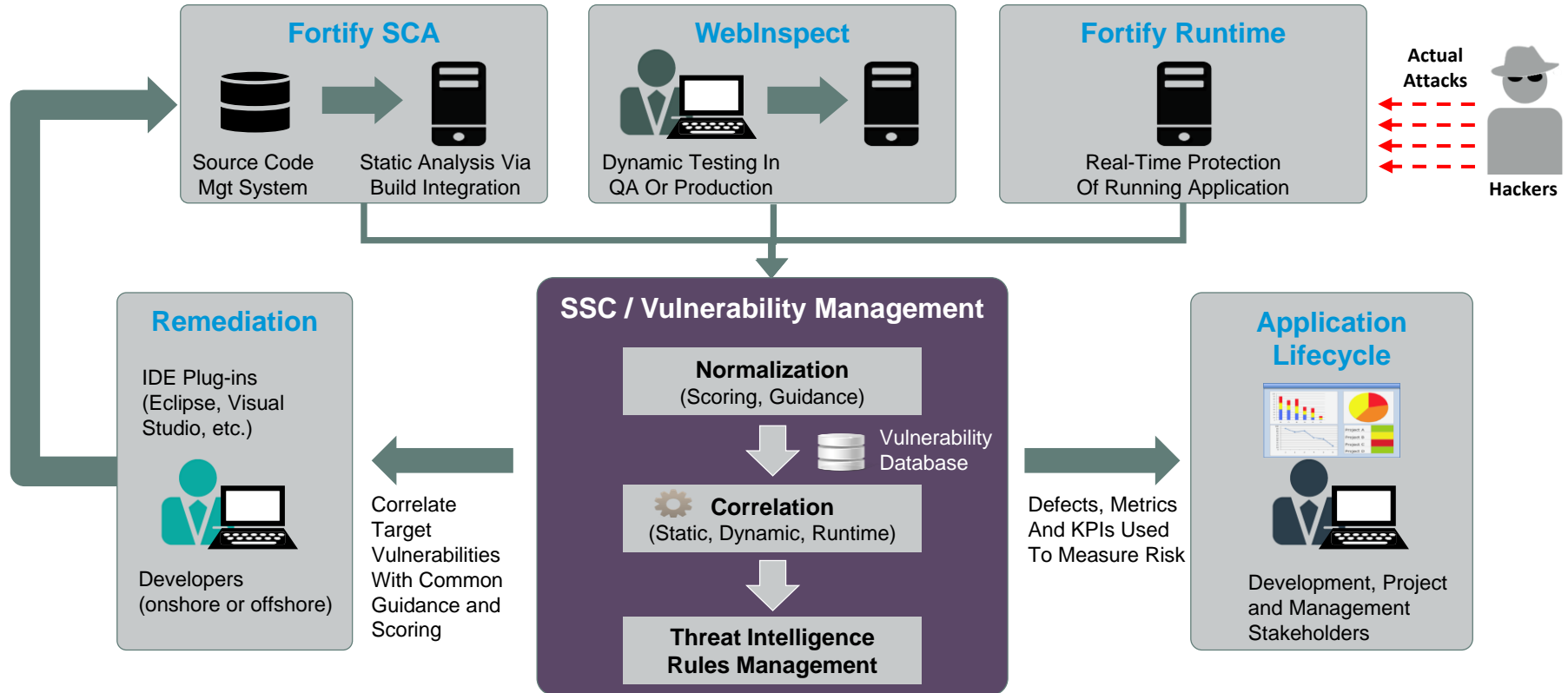
Vulnerability ID	Severity	CVSS	Exploitability	Assigned To			
ForgedPassword.java.191	Career Details	WE	08/30/13	45	Para King	Exploitable	Kan Smith
Encoding.java.754	Career Details	WE	08/30/13	24	Para King	Exploitable	Kan Smith
ChallengeScreen.java.285	Banner Ad Client	SCA	08/30/13	22	Para King	Reliability	Kan Smith
BlindInjection.java.142	Basic Footer	WE	08/30/13	22	Wilmon Wong	Reliability	Kan Smith
ForgedPassword.java.191	Career Details	SCA	08/30/13	45	Wilmon Wong	Not an Issue	Kan Smith
Sandflowletter.pdf	Career Details	SCA	08/30/13	234	Wilmon Wong		Lary Jacobs
BlindInjection.java.142	Career Details	SCA	08/30/13	23	Wilmon Wong		Lary Jacobs
BlindInjection.java.341	Banner Ad Client	SCA	08/30/13	2	Wilmon Wong		Lary Jacobs

Line of code vulnerability detail

Vulnerabilities identified in the scan

Remediation explanation and advice

Fortify Solutions



SSC Functional Areas

– SSC Administration

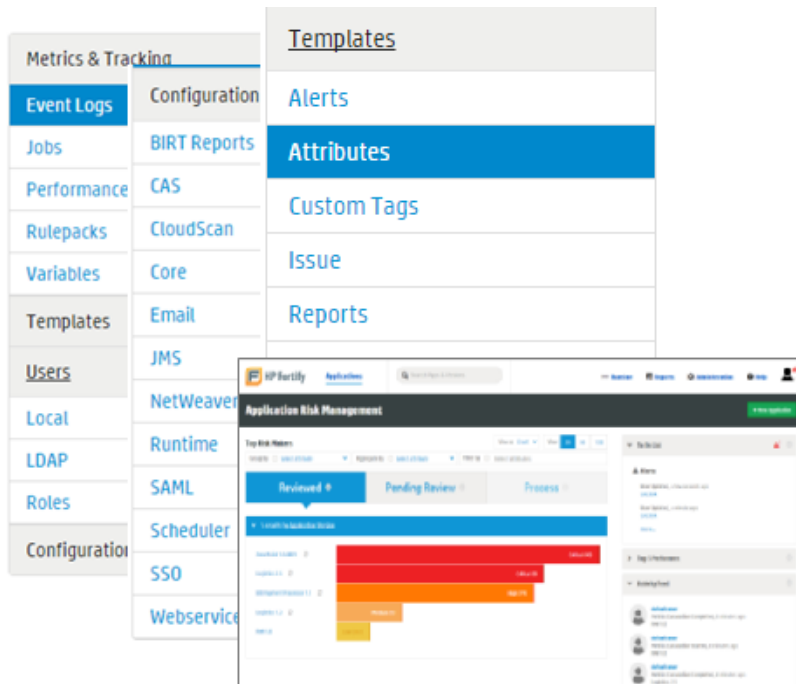
- User Access
- System Configuration
- Issue Template (Project Template)

– Application (Project) Administration

- Application Management
- Attributes Assignment

– Program Management

- Audit Page (Collaboration Module)
- Reporting



Exercise 10:

Software Security Center Walk Through

1. Click on “Launch the Fortify SSC Server”
2. Open a web browser
3. Navigate to <http://localhost:8180/ssc>
4. Login information is in student_logins.txt on your Desktop. Log in as admin.
5. Password is HPpass2016!

SSC Interface

The screenshot displays the HP Fortify Software Security Center (SSC) interface. The browser address bar shows the URL `127.0.0.1:8180/ssc/html/ssc/index.jsp#/dashboard/Chart`. The interface includes a navigation header with the HP Fortify logo, a search bar, and menu items for Applications, Reports, Administration (circled in red), and Help. The main content area is titled "Application Risk Management" and features a "Top Risk Makers" section with a bar chart. The chart shows the following data:

Application	Critical Count
Bill Payment Processor 1.1	Critical (93)
WebGoat.NET v5	Critical (2)
Logistics 2.5	Critical (1)
Logistics 1.3	
RWI 1.0	

Annotations on the image include:

- Administration**: A red line points to the "Administration" menu item in the header.
- Header**: A green line points to the top navigation bar.
- Task List**: A green line points to the "Todo List" and "Alerts" sections on the right.
- Activity List**: A green line points to the "Activity Feed" section on the right.
- Dashboard**: A green line points to the main content area.

Exercise 11

Create a New Project

1. Click on “Launch the Fortify SSC Server”
2. Open a web browser
3. Navigate to <http://localhost:8180/ssc>
4. Login information is in student_logins.txt on your Desktop. Log in as admin.
5. Password is HPpass2016!
6. Click Application
7. Click New Application

New Application

Name: Riches2

Version: v9

Development Phase: New

Create A New Application

The screenshot shows the HP Fortify Applications dashboard. A red line points from the word "Application" to the "Applications" menu item in the top navigation bar. Another red line points from the words "New Application" to the "+ New Application" button in the top right corner of the dashboard. The main content area displays a table of existing applications.

Application	Version	Description	Created
Bill Payment Processor	1.1	Bill payment processing and support interfaces.	02/23/2009
Logistics	1.3	E-commerce web store front with basic shopping card, credit card payment processing and support interface.	05/20/2009
Logistics	2.5	Major updates to backend processing and credit card validation.	04/14/2009
RWI	1.0	Riches Wealth International.	11/23/2009
WebGoat.NET	v5		02/01/2016

Exercise 12

Upload FPR

1. Launch AWB
2. Click Tool
 - a. Click Configure Upload
3. Click Upload Audit Project
4. Enter SSC Login Credentials

Login

SSC URL: <http://localhost:8180/ssc>

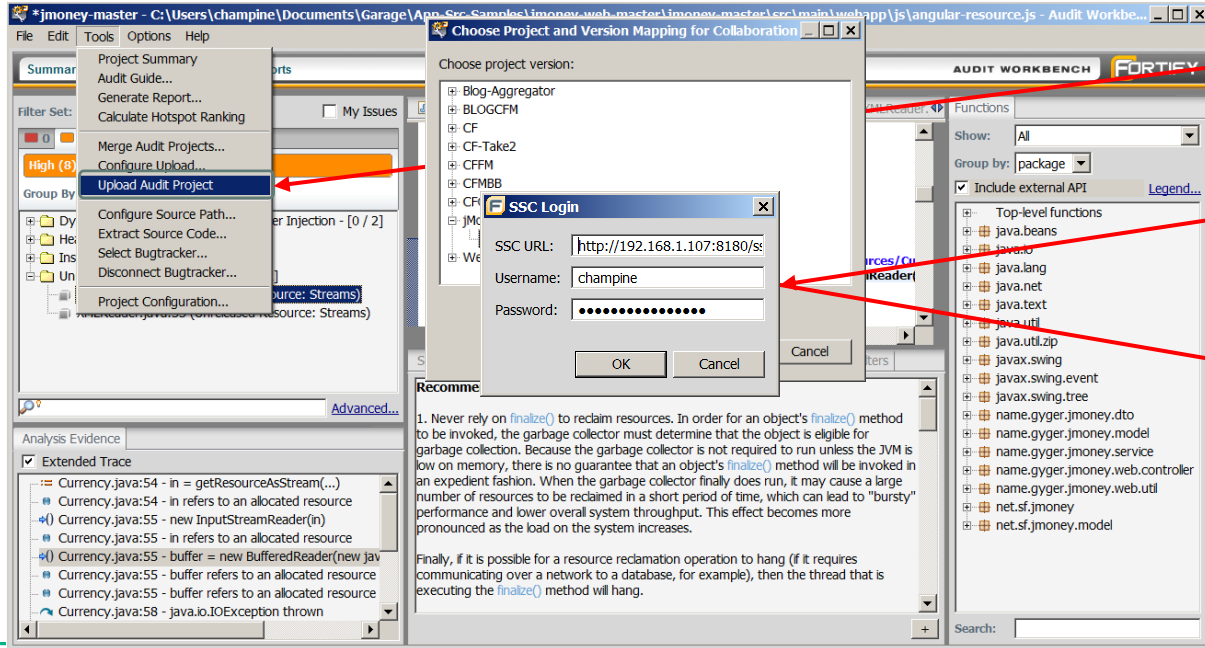
Username: admin

Password: HPpass2016!

Application: Riches2, V9

Upload FPR

– You need to setup [Server Configuration](#) → [Upload FPR Configuration](#) before you can upload FPR



Click Tools then
Upload Audit Project.

Enter SSC user and
password.

Select the appropriate
project and version

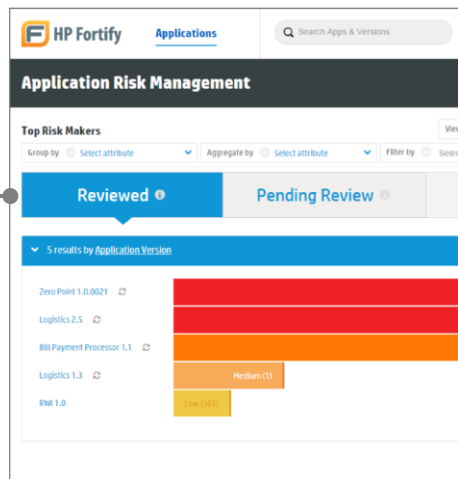


Reporting

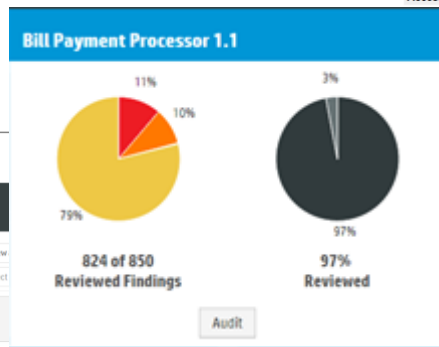
Data Visualization

Dashboards and Reports

Global dashboard highlights risk across software portfolio



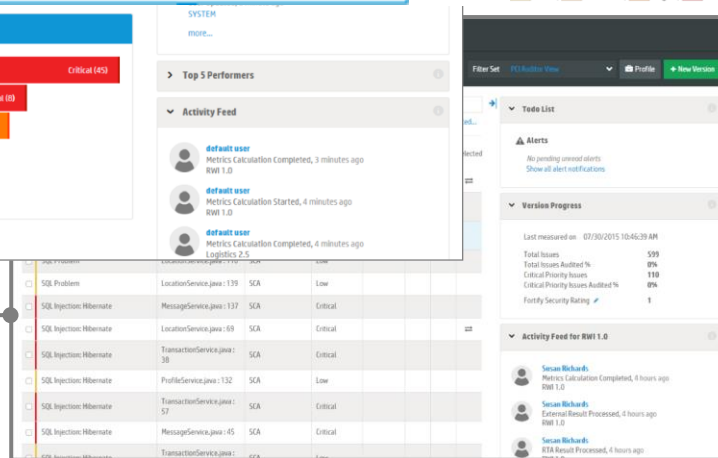
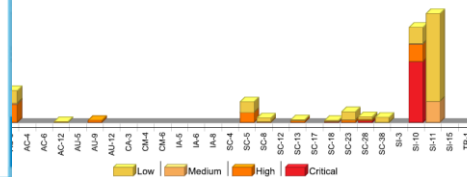
Vulnerability status by application



Remediation Effort (Hrs): 22.6

NIST SP 800-53 Rev.4 groups	Total	Status
Access Control (AC)	52	FAIL
Accountability (AU)	3	FAIL
Configuration Management (CM)	0	PASS
Identification and Authentication (IA)	0	PASS
Assessment and Authorization (CA)	0	PASS
Encryption and Communications Protection (SC)	81	FAIL
System and Information Integrity (SI)	328	FAIL
Privacy (TR)	0	PASS

Issues by NIST SP 800-53 Rev.4 Categories

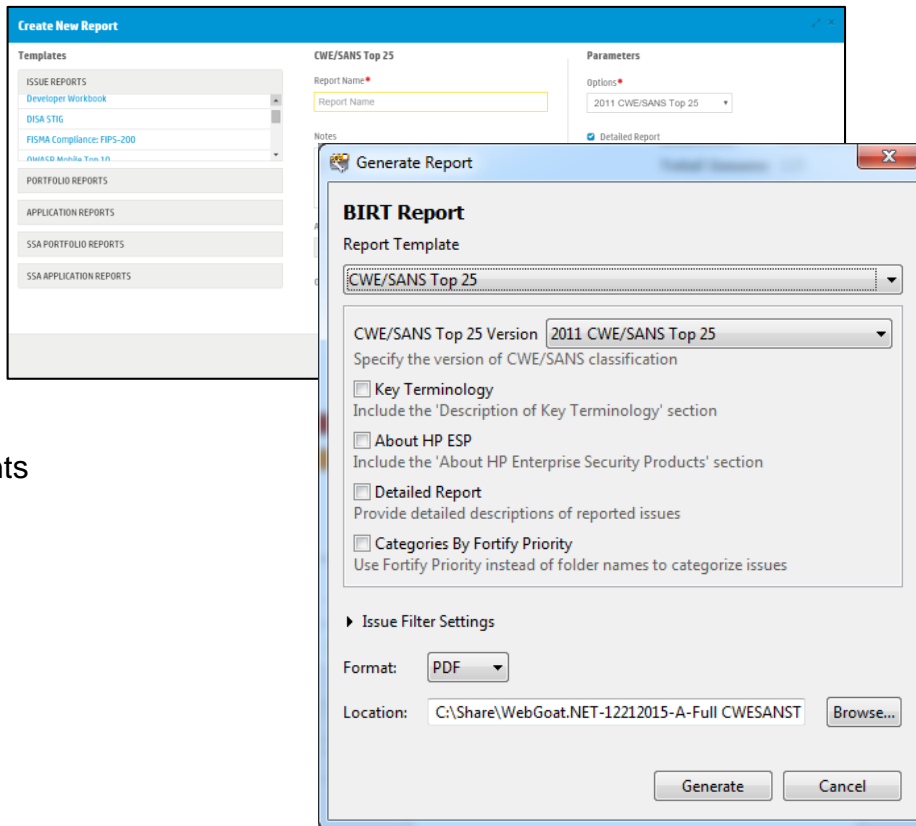


Reporting Features

	AWB/Plugins	SSC
BIRT Reporting	X	X
BIRT Customization		X
Simple Layout Configuration	X	X
PDF	X	X
HTML	X	X
DOC	X	
XLS		X
Asynchronous		X
Synchronous	X	
Issues Reports	X	X
Portfolio Reports		X
Application Reports		X
Dashboards		X

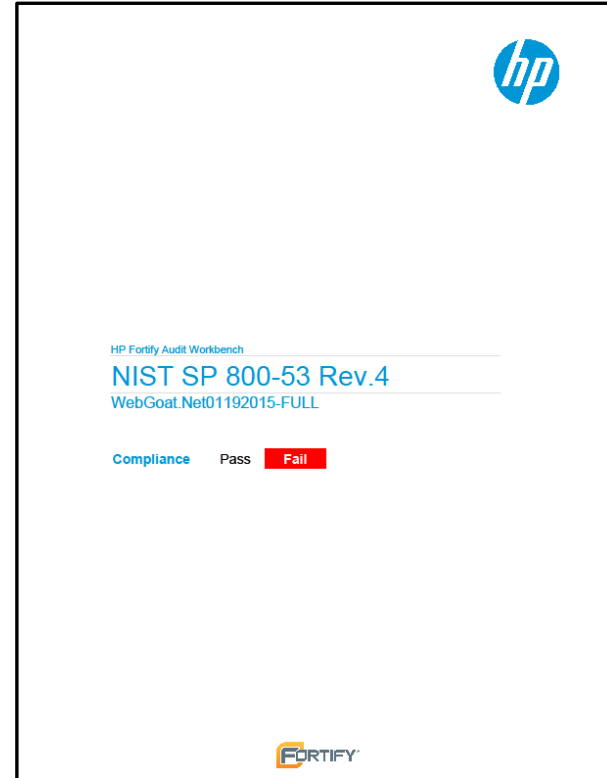
Report Type

- CWE/SANS Top 25
- DISA STIG
- Developer Workbook
- FISMA Compliance: FIPS-200
- OWASP Mobile Top 10
- OWASP Top 10
- PCI DSS Compliance Application Security Requirements



Layout

- Default Report Layout
 - Title Page
 - Table Of Contents
 - Executive Summary
 - Project Description
 - Issue Breakdown
 - Issue Detail/Summary



Exercise 12

Generate AWB Reports

1. Launch AWB
2. Click Reports
3. Generate BIRT Report - Security Auditor View
4. Click Reports
5. Generate BIRT Report – Quick View
6. Click Reports
7. Click Tools->Generate Legacy Report

Features

- New BIRT Reporting Engine
- Simple Layout Configuration
- Saves as DOC,HTML, PDF
- Synchronous

AWB Reports: Generation

Reports

The screenshot shows the Audit Workbench (AWB) interface for a project named "WebGoat.Net01192015-FULL". The "Reports" menu item is highlighted with a red circle. The interface is divided into several panes:

- Summary Pane:** Shows project details including Build ID (WebGoat.Net01192015-FULL), Scan Date (Jan 19, 2016), Scanned files (309 files, 4,517 LOC), Total Issues (466), and Certification (Results Certification Valid).
- Issues Summary:** A bar chart titled "All Issues by Folder" showing 8 Medium issues and 41 Critical issues.
- Issue List:** A list of issues with a filter set to "Security Auditor View". It shows 43 Critical and 2 Suppressed issues. The list includes items like "SC-28 Protection of Information at Rest (P1) - [0 / 1]" and "SI-10 Information Input Validation (P1) - [3 / 40]".
- Functions Pane:** A tree view of scanned files and folders, including "ASP", "DotNetGoat", "log4net", and various system and application-specific folders.

The Windows taskbar at the bottom shows the time as 1:56 PM on 2/19/2016.

Exercise 13

Generate SSC Reports

1. Click on “Launch the Fortify SSC Server”
2. Open a web browser
3. Navigate to <http://localhost:8180/ssc>
4. Login information is in student_logins.txt on your Desktop. Log in as admin.
5. Password is HPpass2016!
6. Click Reports
7. Click New Report

Features

- New BIRT Reporting Engine
- BIRT Customizations
- Simple Layout Configuration
- Saves as XLS,HTML, PDF
- Asynchronous
- Dashboard Portfolio and Application Reports

Generate SSC Reports

The screenshot shows the HP Fortify Reports interface. A red circle highlights the 'Reports' menu item in the top navigation bar. Another red circle highlights the '+ New Report' button in the top right corner of the Reports section. A red arrow points from the text 'New Report' on the right side of the image to this button. The main content area displays two tables: 'Issue Reports' and 'Portfolio Reports'.

Report Name	Type	Date	Created By	Status	Notes	Versions
> SWE	CWE/SANS Top 25	02/04/2016 2:38:40 PM	User, Default	Processing Complete		WebGoat.NET v5

Report Name	Type	Date	Created By	Status	Notes	Versions
> cccc	Issue Trending	02/04/2016 3:30:32 PM	User, Default	Processing Complete		Bill Payment Processor 1.1

New Report



Q&A?



**Hewlett Packard
Enterprise**

HPE WebInspect Hands on Workshop

Haleh Nematollahy
Jeffrey Hsiao



Agenda - Dynamic

- Overview of scanning workflow
- Scan Riches application
- Authentication
- Scanning policies
- Reviewing scan results
- Reporting
- Overview of WebInspect Enterprise

The Solution



HPE Fortify helps you protect your applications



Application assessment

Assess

Find security vulnerabilities in any type of software



In-house
Outsourced
Commercial
Open source

Software security assurance

Assure

Fix security flaws in source code before it ships



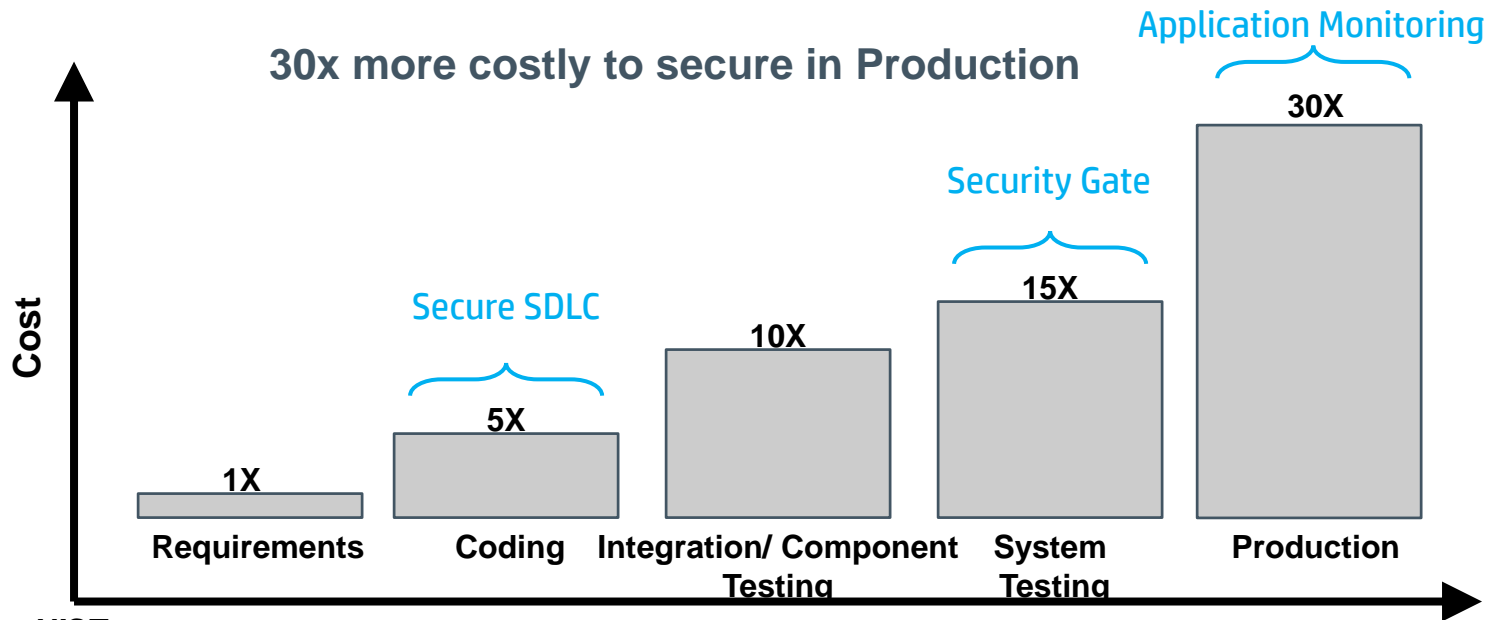
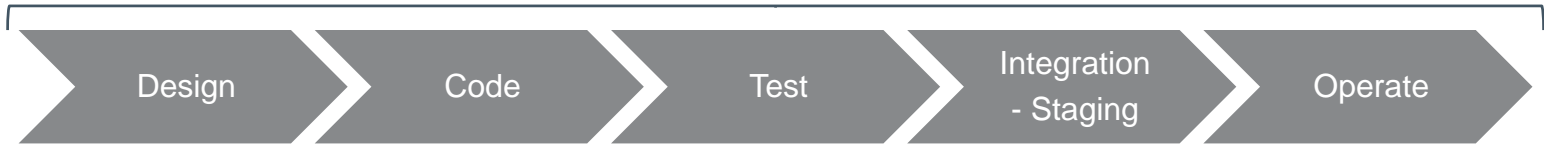
Application protection

Protect

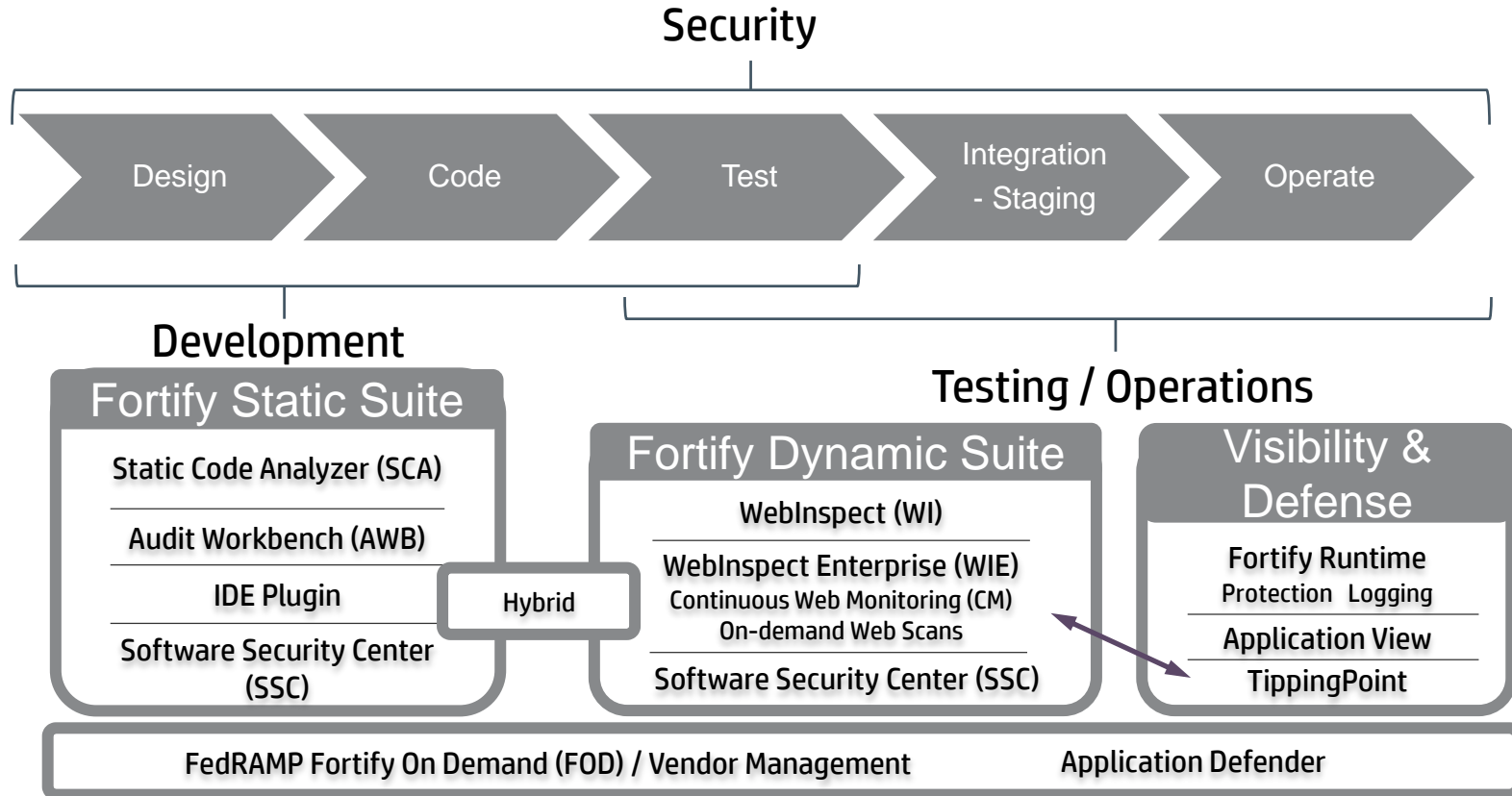
Fortify applications against attack in production

Software Security Assurance (SSA & SDLC)

Security



Software Security Assurance (SSA & SDLC)





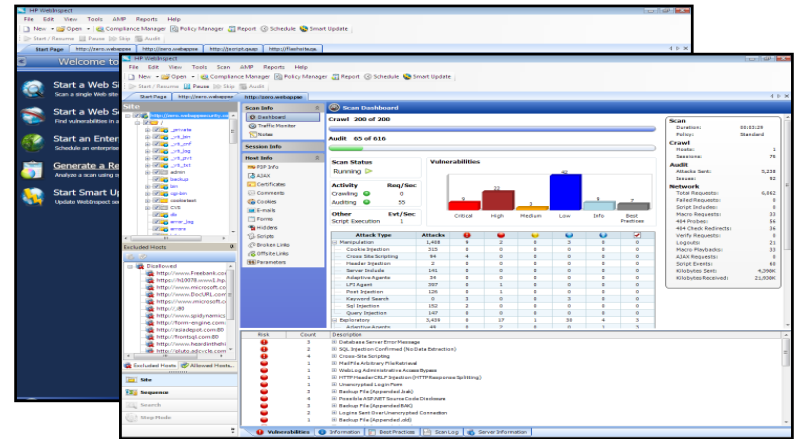
WebInspect Dynamic Analysis

HPE WebInspect

Dynamics analysis – find critical security issues in running applications

– Features:

- Quickly identify risk in existing applications
- Automate dynamic application security testing of any technology, from development through production
- Validate vulnerabilities in running applications, prioritizing the most critical issues for root-cause analysis



Included In Every WebInspect License

- **SmartCard** / CAC Authentication
- FISMA / 800-53 **Compliance Reporting**
- Scan **Web Applications, SOAP and RESTful** Services, URL Rewriting
- Scan Mobile Web sites, plus Mobile Native Scan
- Advanced Crawler with Javascript execution
- Integration into **ArcSight, Tipping Point, WAFs, Software Security Center, WebInspect Enterprise**
- Hybrid scanning with the **WebInspect Agent**
- Tools for manual Testing and Penetration including automatic SQL Injection
- WebInspect API plus **BURP Integration**
- SmartUpdate automatic frequent security content updates from the largest dedicated Software Security Research group.
- **OFFLINE** activations and updates

HPE Fortify Hybrid Analysis

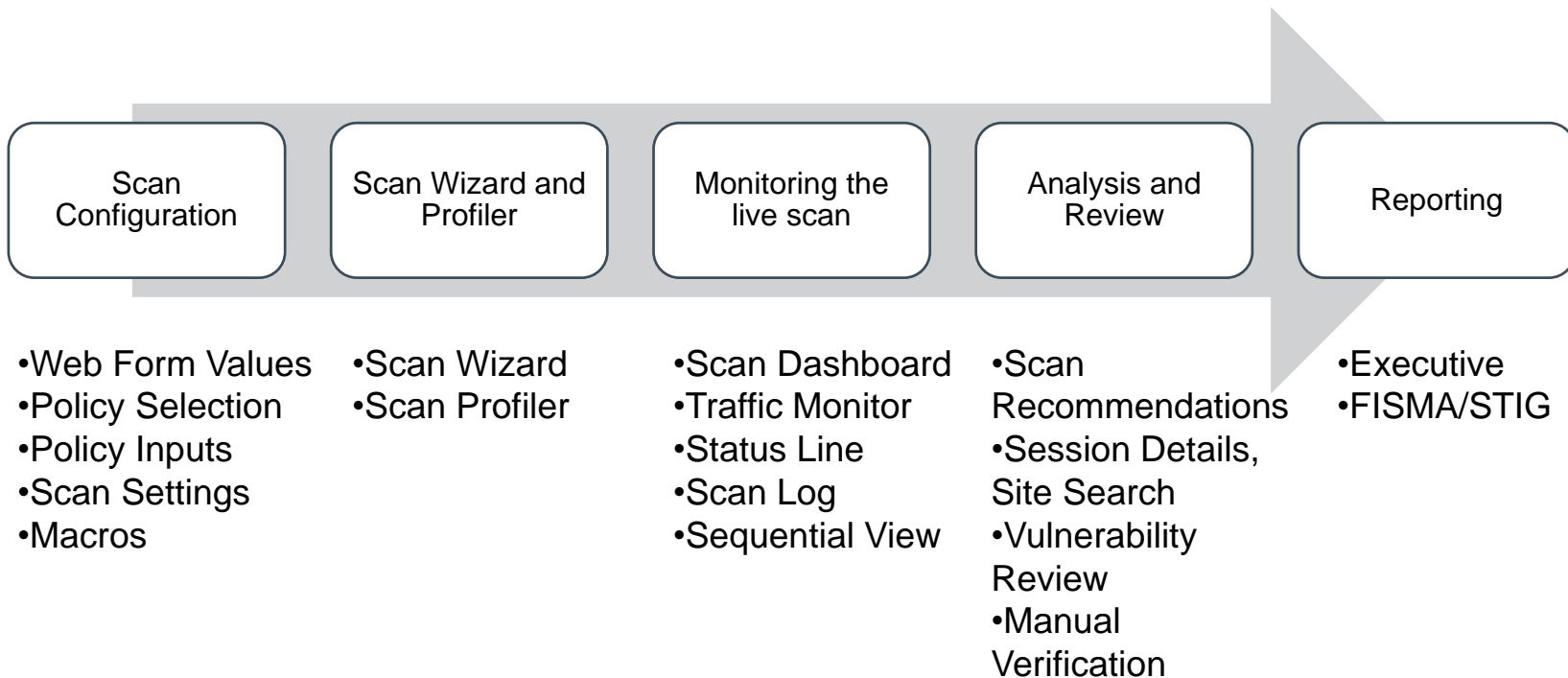
WebInspect Agent

- Provides gray-box testing capability
 - Fix faster: Provide line of code details/ Stack Trace
 - Find more: Enables deeper, more thorough penetration testing
- Complements static analysis
 - Provides outside-in perspective
 - Validates findings
 - Focuses developer attention on exploitable vulnerabilities
- Enables correlation between static and dynamic analyses
 - Follow exploit all the way to the line of code
 - Out of the box integration – no additional customization required



Run a WI Scan

High Level Workflow



Exercise 1

Scan Riches – Guided Unauthenticated Scan



- Select your Template
- Verify <http://127.0.0.1:8080/riches/>
- Restrict it to this directory and its subs
- Explore each pane
- Keep Defaults
- Use Riches Optimization
- Enhance Coverage If you wish.
- There are no False Positives to import.
- Turn on Traffic Monitor.
- Save as scan template for later use
- **Start Scan**

Predefined Templates

- Create a Standard Web Site Scan**
Default scan settings are designed to focus more on coverage than performance. Larger sites could take days to crawl with these settings.
- Create a Quick Web Site Scan**
A scan that focuses on breadth and performance rather than digging deep. Especially good for very large sites.
- Create a Thorough Web Site Scan**
Thorough scan settings are designed to perform an exhaustive crawl of your site. It is recommended that you split your site up into parts and only scan smaller chunks of your site with these settings. Not recommended for large sites.

Recent Templates

- tomato_cart
- katie
- Riches_Demo
- Default
- Factory Default
- From File ...

Recent Scans

Scan Now

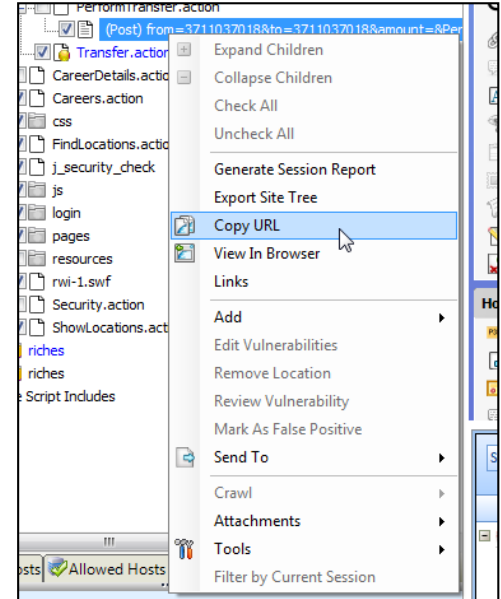
Here is a summary of the scan that you are about to start.

Scan Name: Site:
URL: <http://127.0.0.1:8080/riches/>
Uses Authentication: Login Macro
Policy: Criticals and Highs
Client: Firefox

Start Scan

While This Scan is Running

- Notice that vulnerabilities are reported even while still crawling
- Review the growing Site Tree
 - Pause, review **Sequence View**, **Search View**, and **Step Mode**
 - Explore the **Context Menu in the Site Tree**
- Traffic Monitor
- **Add the column “Location”** and move up
- Explore HTTP Packets
- Explore Host Info



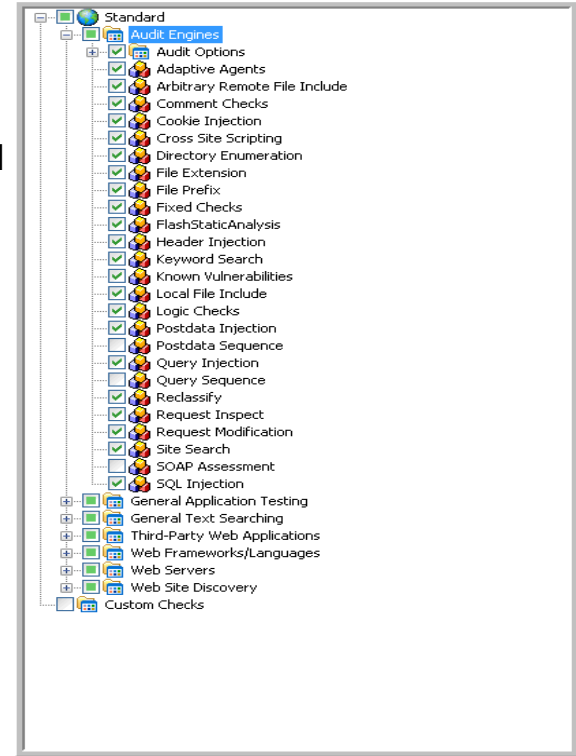
We can leave this scan running, and open a completed scan in a new tab
You can have 2 scans running at a time



Policies and Other Important Settings

Policy Manager Walkthrough

- A Policy is a collection of tests in the SecureBase to be executed.
- Several Default policies exist – they cannot be modified but can be inherited
- You can select and deselect at any level.
- Checks require corresponding Engine
- Custom Checks – create custom checks in different engines.
- Default policy is “Standard”



Exercise 2

How to Create a Policy



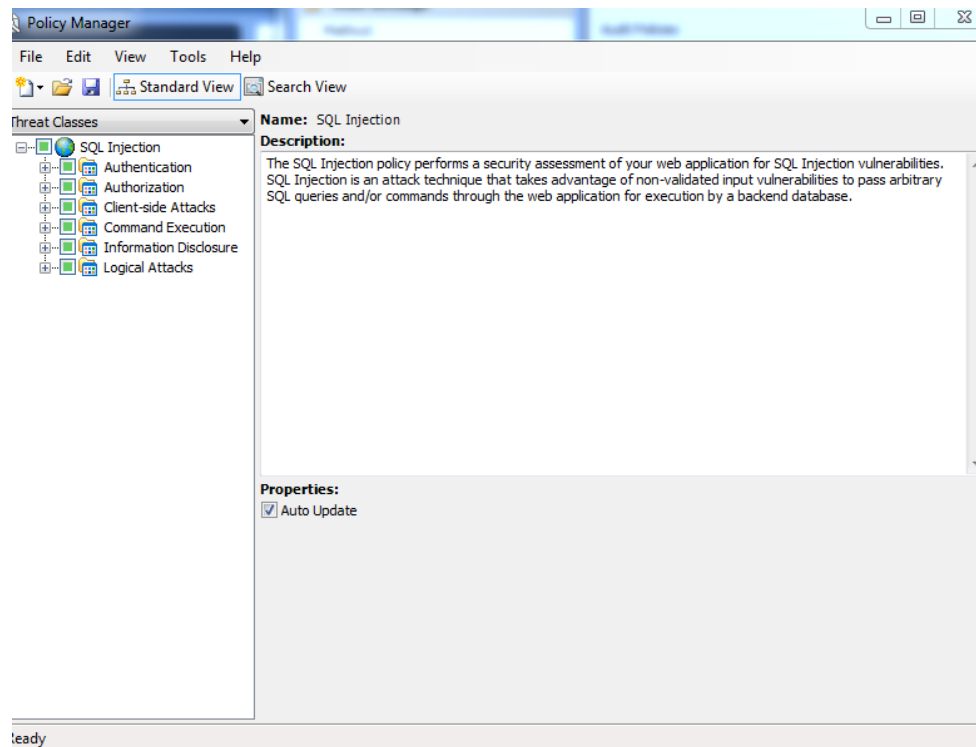
- Click **Create** and The Policy Manager tool opens.
- Select **New** from the **File** menu (or click the New Policy icon).
- Select the policy on which you will model a new one from:
 - SQL Injection Policy → Select Command Injection → SQL Injection → Site Database Disclosure and SQLI
- Save
- Under Custom → My Policy
- When you configure your settings, pick your new My Policy for scan

Exercise 2 Continued

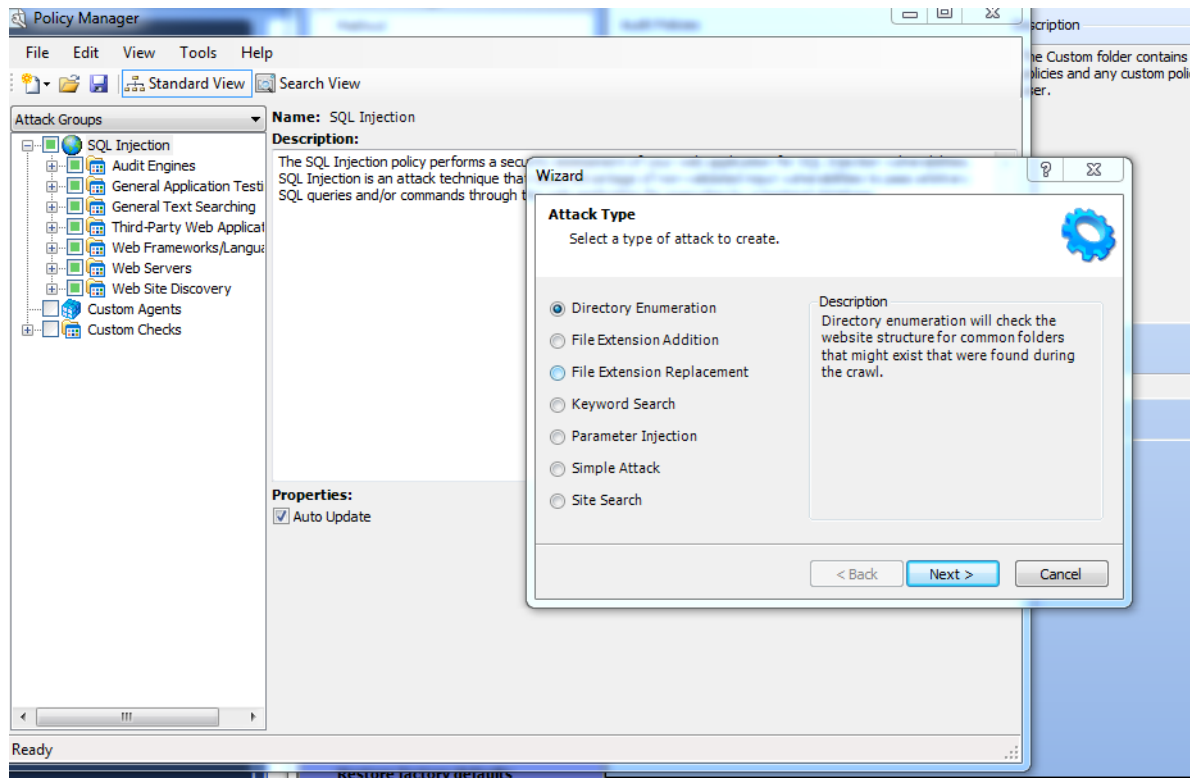


- Do a New Critical and Highs Policy
 - Default Settings → Policy → Create → File → New → Criticals and Highs Policy
 - Search for and deselect all checks mapped to CWE 521
 - Search View → Criteria → Vulnerability ID = CWE ID Contains 521
 - Deselect All
 - Save the new Custom Policy MYCWEPOLICY

How to Create a Policy

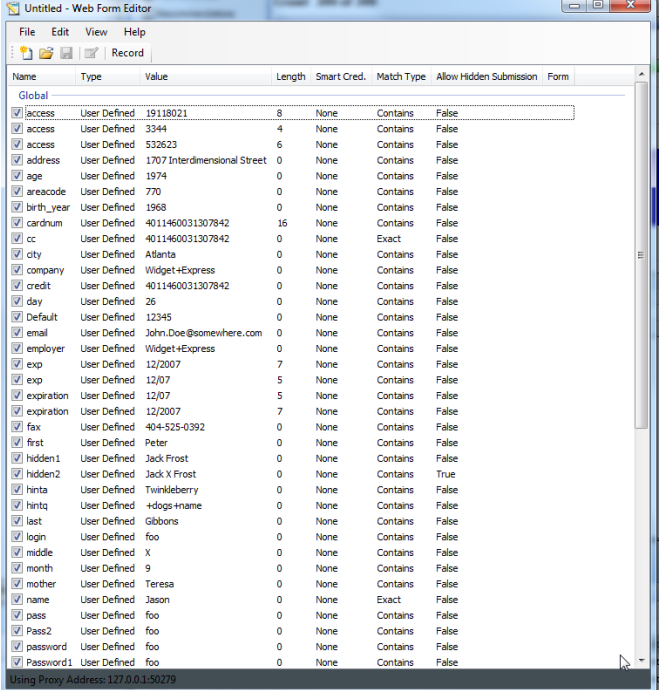


How to Create a Policy



WebForm Values

- Directly enter values or record them.
- Tag values as “manual input” prompting you for real time values
 - Captchas
 - One-Time Pins
 - Disposable values
- Mutiple Values based on Field Width
- Values can be global or per URL
- Modify some values and save your file

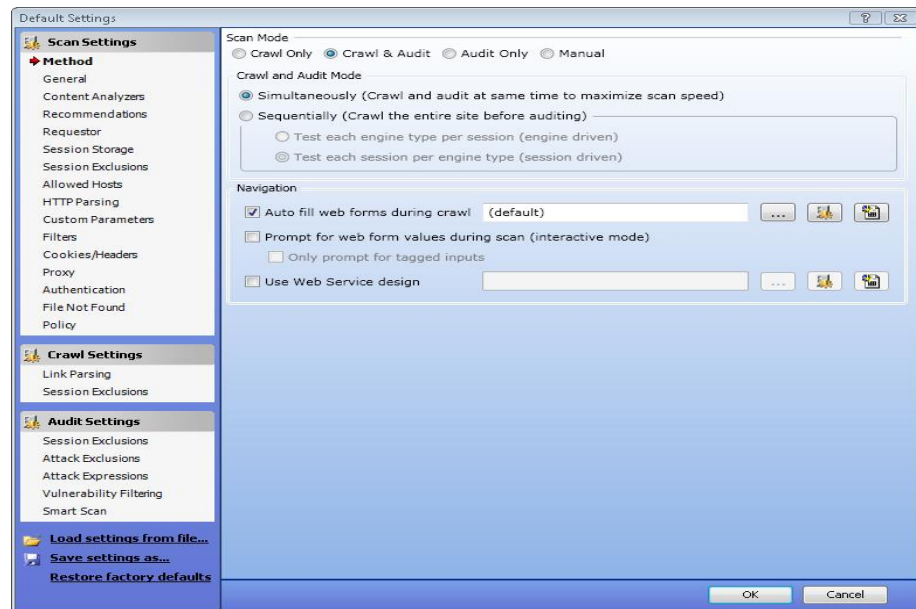


The screenshot shows the 'Web Form Editor' window with a table of form fields. The table has columns for Name, Type, Value, Length, Smart Cred., Match Type, Allow Hidden Submission, and Form. The 'Global' section is selected, and the 'access' field is highlighted.

Name	Type	Value	Length	Smart Cred.	Match Type	Allow Hidden Submission	Form
access	User Defined	19118021	8	None	Contains	False	
access	User Defined	3344	4	None	Contains	False	
access	User Defined	532623	6	None	Contains	False	
address	User Defined	1707 Interdimensional Street	0	None	Contains	False	
age	User Defined	1974	0	None	Contains	False	
areacode	User Defined	770	0	None	Contains	False	
birth_year	User Defined	1968	0	None	Contains	False	
cardnum	User Defined	4011460031307842	16	None	Contains	False	
cc	User Defined	4011460031307842	0	None	Exact	False	
city	User Defined	Atlanta	0	None	Contains	False	
company	User Defined	Widget+Express	0	None	Contains	False	
credit	User Defined	4011460031307842	0	None	Contains	False	
day	User Defined	26	0	None	Contains	False	
Default	User Defined	12345	0	None	Contains	False	
email	User Defined	John.Doe@somewhere.com	0	None	Contains	False	
employer	User Defined	Widget+Express	0	None	Contains	False	
exp	User Defined	12/2007	7	None	Contains	False	
exp	User Defined	12/07	5	None	Contains	False	
expiration	User Defined	12/07	5	None	Contains	False	
expiration	User Defined	12/2007	7	None	Contains	False	
fax	User Defined	404-525-0392	0	None	Contains	False	
first	User Defined	Peter	0	None	Contains	False	
hidden1	User Defined	Jack Frost	0	None	Contains	False	
hidden2	User Defined	Jack X Frost	0	None	Contains	True	
hinta	User Defined	Twinklberry	0	None	Contains	False	
hintq	User Defined	+dogs+name	0	None	Contains	False	
last	User Defined	Gibbons	0	None	Contains	False	
login	User Defined	foo	0	None	Contains	False	
middle	User Defined	X	0	None	Contains	False	
month	User Defined	9	0	None	Contains	False	
mother	User Defined	Teresa	0	None	Contains	False	
name	User Defined	Jason	0	None	Exact	False	
pass	User Defined	foo	0	None	Contains	False	
Pass2	User Defined	foo	0	None	Contains	False	
password	User Defined	foo	0	None	Contains	False	
Password1	User Defined	foo	0	None	Contains	False	

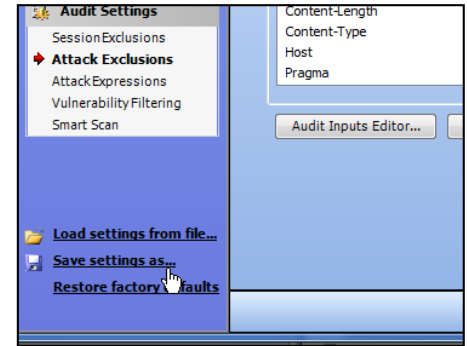
Settings

- Method: Auto Fill Web Forms – select our Web Form Values
- General: Enable Traffic Monitor
- General: Limit Maximum Web Form Submission To: 10
- General: Perform redundant page detection
- Content Analyzers: Review Javascript
- Proxy
- Authentication
- File Not Found Settings



Settings

- Change the following Settings:
 - Method: Auto Fill Web Forms – select our Web Form Values
 - General: Enable Traffic Monitor
 - General: Limit Maximum Web Form Submission To: 10
 - General: Perform redundant page detection
 - Authentication: Select your Login Macro
 - Policy: Specify your Custom Policy
-
- Save the Settings – you now have your entire configuration stored in a reusable format.





Authentication



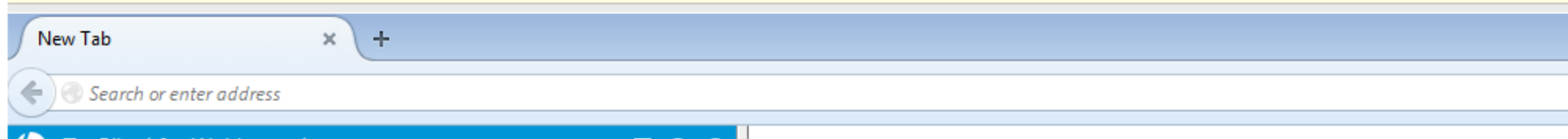
Exercise 3

Riches Login Macro

- Go To Tools and open Login Macro Recorder
- Follow the Yellow-Bar Prompts to create a login to the Riches bank site using **Eddie / Eddie** as user/Pass
- Go ahead and test your Macro

Click Record, navigate to your site and then log in

Record



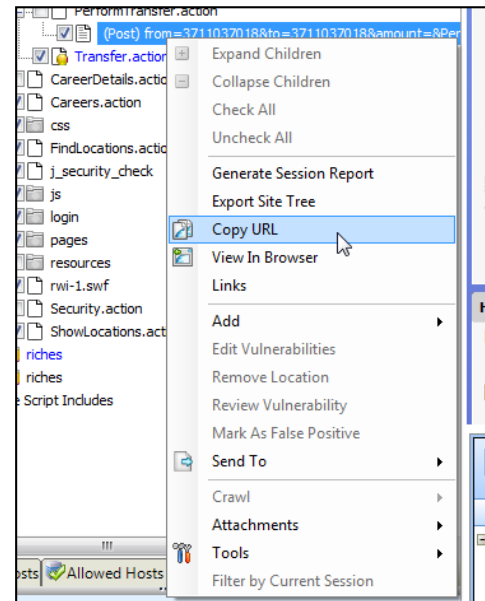


Auditing Results

While This Scan is Running



- Notice that vulnerabilities are reported even while still crawling
- Review the growing Site Tree
 - Pause, review **Sequence View**, **Search View**, and **Step Mode**
 - Explore the **Context Menu** in the Site Tree
- Traffic Monitor
- **Add the column “Location”** and move up
- Explore HTTP Packets
- Explore Host Info

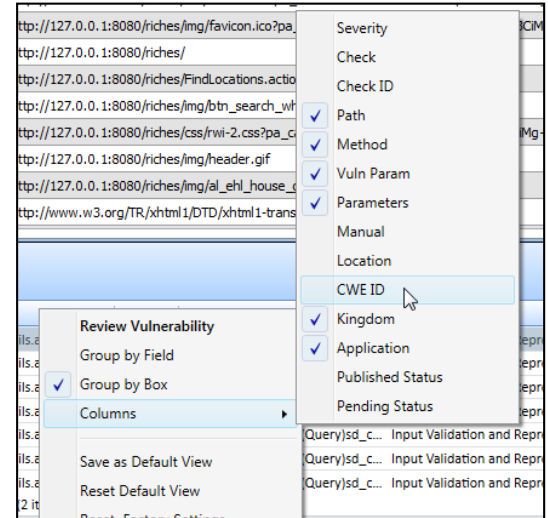


We can leave this scan running, and open a completed scan in a new tab

You can have 2 scans running at a time

Review a Completed Scan

- Review the Site Tree
- Vulnerability Pane:
 - Sort and group by Duplicates, Severity
 - **Select column “CWE ID”**
 - Review packets for a vulnerability
 - Change severity for a vulnerability
 - Add notes to a vulnerability
 - Review Steps
- Retest a single vulnerability / All Vulnerabilities/Repeat Scan
- Add your own manual finding (Edit Vulnerabilities)



Exercise 4



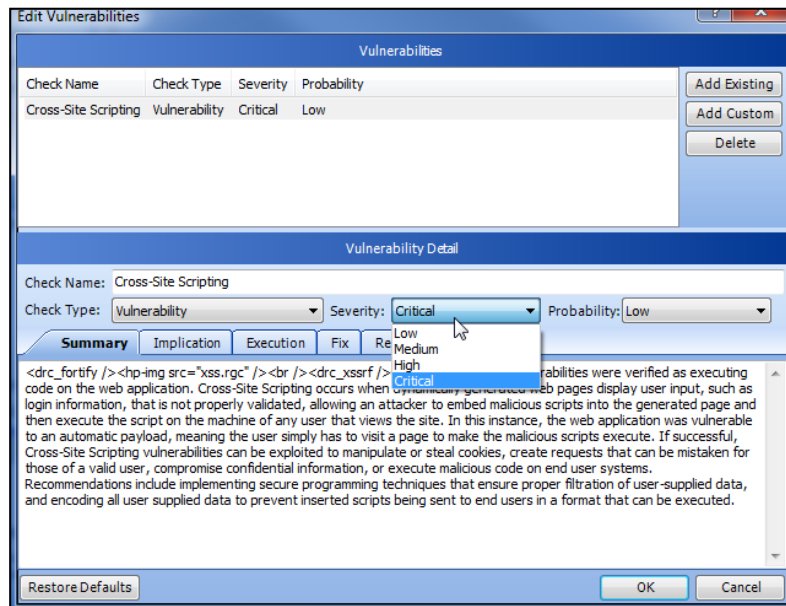
Validate a finding

- Look at request, response, vulnerability description and stack trace
- Retest the vulnerability
- Look at one finding
- Select “Edit Vulnerability” and change severity for one
- Add your own information to the reporting.
- Find one false positive and mark it as such

Edit a Vulnerability



- Right Click a Vulnerability in the Site Tree
- Select “Edit Vulnerability”
- Change the Severity
- Add your own information to the reporting.
- This is all saved to the scan file
- Flows upstream with scan:
 - Reports,
 - FPR’s into Software Security Center,
 - Scan uploads into WebInspect Enterprise



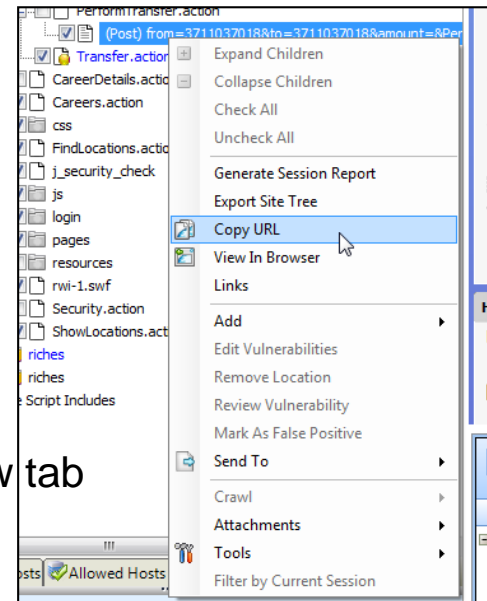
While This Scan is Running



- Notice that vulnerabilities are reported even while still crawling
- Review the growing Site Tree
 - Pause, review **Sequence View**, **Search View**, and **Step Mode**
 - Explore the **Context Menu in the Site Tree**

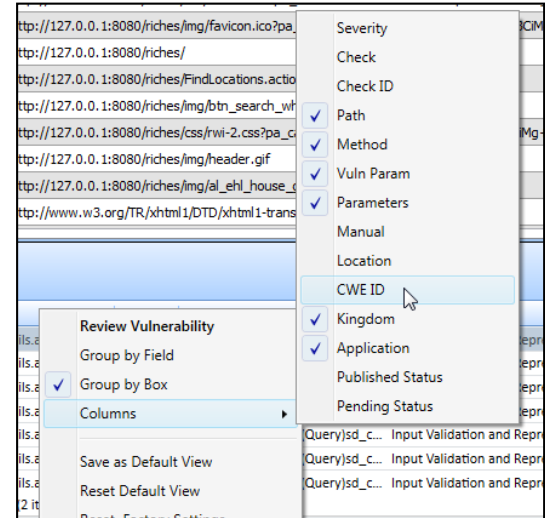
We can leave this scan running, and open a completed scan in a new tab

You can have 2 scans running at a time



Review a Completed Scan

- Review the Site Tree
- Vulnerability Pane:
 - **Select column “CWE ID”**
 - Review packets for a vulnerability
 - Change severity for a vulnerability
 - Add notes to a vulnerability
 - Review Steps
- Retest a single vulnerability / All Vulnerabilities/Repeat Scan
- Add your own manual finding (Edit Vulnerabilities)





Reports and Data Export

Exercise 5

Generate a Report



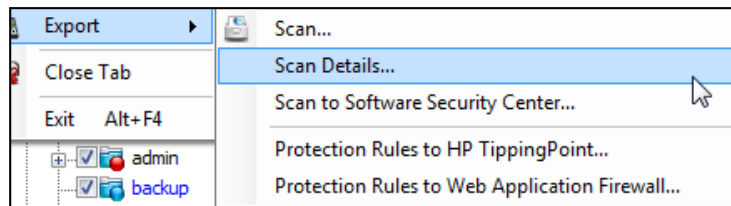
- Use your existing scan
- Select Executive Summary, Compliance and Vulnerability
- For Compliance: Select DoD Application Security And Development STIG V3 R9 or FISMA.
- For Executive Summary: No change
- For Vulnerability: Clear ALL Severities except Critical.

Exercise 6

Export Data



- Scan
 - exports to proprietary binary .scan format
 - saves entire scan and be re-loaded into WebInspect
- Scan Details
 - Exports selectable sections or full scan
 - XML based - can be used for integration (STIG Viewer)
- Scan to Software Security Center
- Saves results in FPR format for uploading into Software Security Center to be managed alongside static scans





Fortify and SSC Demo

Upload Results to SSC from WI



WebInspect Enterprise

HPE WebInspect Enterprise

Extending effective application security testing across the entire enterprise

– Problem it solves:

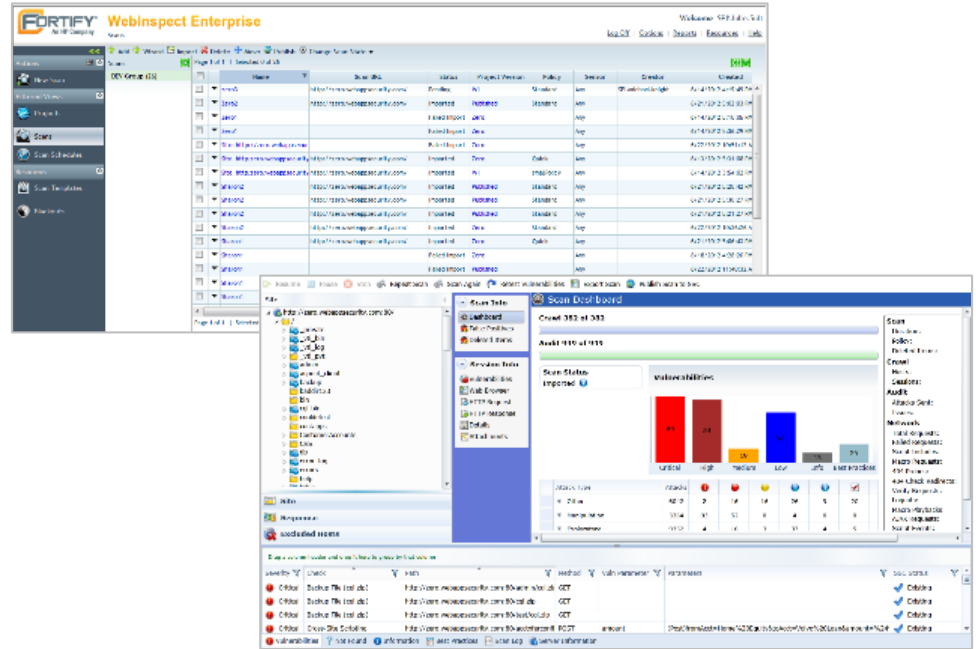
- Manages large-scale, distributed security testing programs across thousands of applications

– Features:

- Monitor critical metrics, progress and trends across large-scale application security testing programs
- Provide an ongoing enterprise-wide view of production and pre-production application security assurance
- Control your application security program through role-based scanning and reporting administration

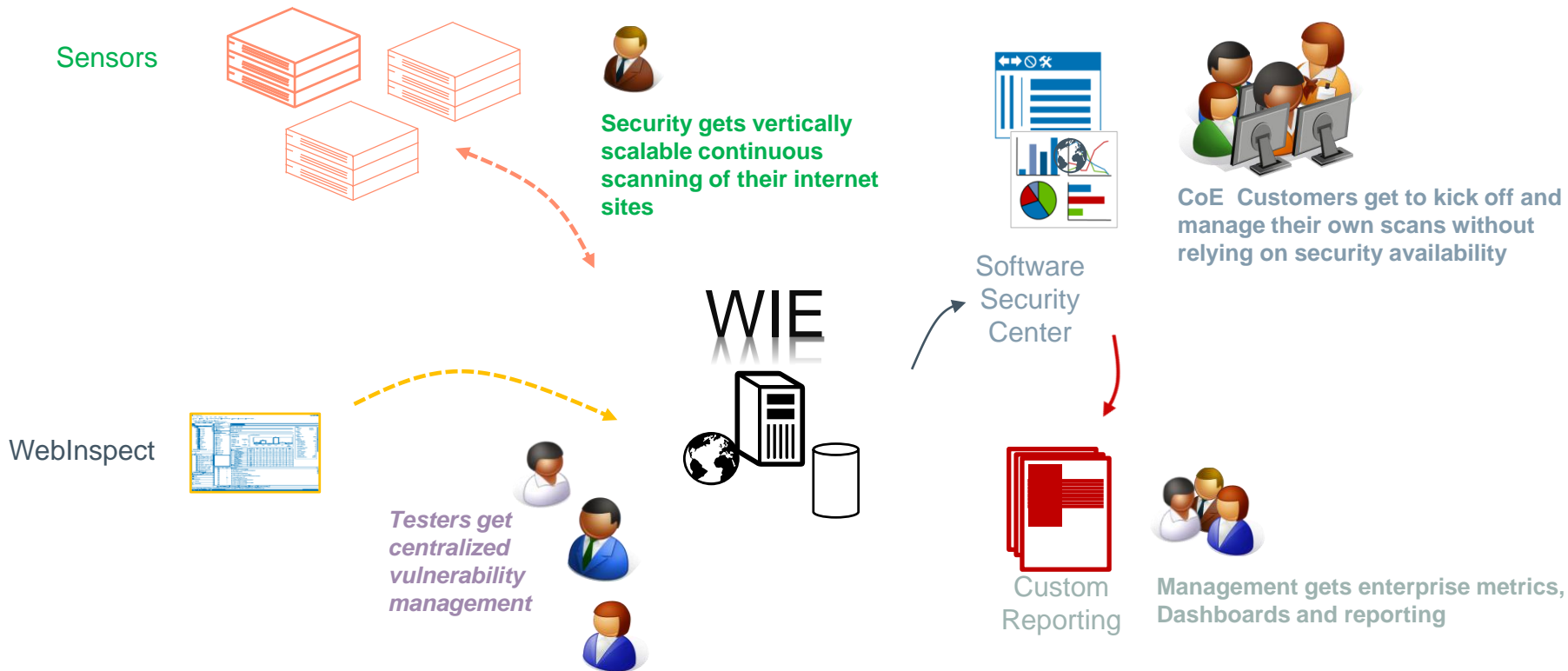
– Benefits:

- Eliminate inefficient and inconsistent assessment and vulnerability management processes
- Increase visibility and control of security testing efforts and reporting
- Prove compliance with regulations, standards and policies



WIE Architecture

Flexible modeling for Web App Continuous Monitoring, Centralized Vulnerability Management, and COE



WebInspect Enterprise Web App Monitoring

- Continuous Web Application Monitoring
- Schedule automated scans
 - Schedule regular, repeatable assessments of your applications
 - Schedule scans during ‘quiet’ periods
 - Adjustable policy controls scan depth and breadth
- Compare Results over Time
 - Highlight new issues and threats
 - Track application risk improvements
- Notifications for management
 - Scan or Update completions or errors
 - System trouble
- On-Demand Scanning
- Request automated scans
 - Scan applications as requested by development in testing and UAT env.
 - Provides developers and IA with access to dynamic scan results before C&A
 - Allows teams to track security posture across milestones and sprints

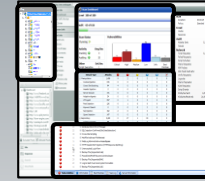


WebInspect Enterprise For Continuous Monitoring



Enterprise Sensors

Deployable Scanning Engines allows scalable, constant validation of operating sites



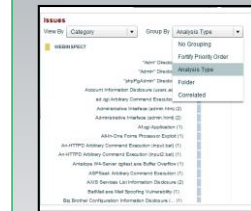
Browser-Based Scanning

Easily share testing capabilities with development groups without having to provision and manage software



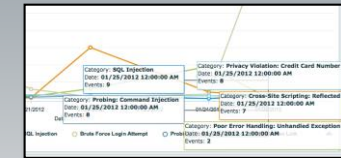
WebInspect

IV&V and ST&E teams can continue using WebInspect on a stand-alone basis while still leveraging the centralized artifact and vulnerability management



Fastest Road to "Fixed"

Developers get direct access to all scan results for quickest path to remediation



Central Reports and Dash

Full reporting, KPI and management metrics available across all applications.

WebInspect Enterprise: Centralized Scan Results

The screenshot displays the Fortify WebInspect Enterprise interface. The top left shows the Fortify logo and 'WebInspect Enterprise' title. The top right has a welcome message for 'WIN-G4SKN8G01E\admin' and navigation links like 'Log Off', 'Options', 'Resources', and 'Help'. Below this is a 'Scans' section with a table of scan results.

Name	Scan URL	Status	Project Version	Policy	Sensor	Creator
CM - WWW Main Site - 2013-01	http://zero.webappsecurity.com/	Failed	extranet.agency.mil	Standard	Any	
Site: http://zero.webappsecu	http://zero.webappsecurity.com/	Pending	extranet.agency.mil	Standard	Any	WIN-G4SKN8G01E
List-driven:	http://zero.webappsecurity.com/	Pending	extranet.agency.mil	Quick	Any	WIN-G4SKN8G01E
CM - WWW Main Site - 2012-12	http://zero.webappsecurity.com/	Pending	extranet.agency.mil	Standard	Any	
Site: http://zero.webappsecu	http://zero.webappsecurity.com/	Pending	extranet.agency.mil	Standard	Any	
November scan - extranet.aga	http://zero.webappsecu					
Copy of Site: http://zero.web	http://zero.webappsecu					
www.Agency.Mil	http://zero.webappsecu					
Site: http://zero.webappsecu	http://zero.webappsecu					
Site: http://zero.webappsecu	http://zero.webappsecu					

The right side of the interface shows a 'Scan Visualization - Site: http://zero.webappsecurity.com/' window. It displays a 'Session HTTP Response' for the URL 'http://zero.webappsecurity.com:80/bank/account-activity.html?accountId=1%29%3...'. The response contains JavaScript code for footer links and a click handler.var footerLinks = {
 "download_webinspect_link": { absolute: true, page: "https://download.hpsmartupdate.com/webinspect/" },
 "contact_hp_link": { absolute: true, page: "https://support.fortify.com" },
 "privacy_statement_link": { absolute: true, page: "http://www8.hp.com/us/en/privacy/privacy.html" },
 "terms_of_use_link": { absolute: true, page: "http://www8.hp.com/us/en/privacy/terms-of-use.html" }
};

\$.each(footerLinks, function(linkId, link) {
 attachClickHandler('span[id="'+ linkId + '"]', function(event) {
 event.preventDefault();
 if (link.absolute) {
 window.location.href = link.page;
 } else {
 window.location.href = path + link.page + ".html";
 }
 });
});
</script>
</body>
</html>

Below the response, a table of scan results is shown, grouped by severity. The table includes columns for Path, Method, Vuln Parameter, Parameters, SSC Publish Status, SSC Status, Stack Trace?, and Application.

Path	Method	Vuln Parameter	Parameters	SSC Publish Status	SSC Status	Stack Trace?	Application
Critical : Items (32)							
Buffer Overflow : Items (1)							
http://zero.webappsecurity.com:80/cgi-bin/post-	GET			Existing	Existing	No	
Cross-Site Scripting: Reflected : Items (23)							
http://zero.webappsecurity.com:80/bank/account	GET	accountId	{Query}accountId=1%29%3b%61	Existing	Existing	No	
http://zero.webappsecurity.com:80/bank/account	GET	accountId	{Query}accountId=1%29%3b%61	Existing	Existing	No	
http://zero.webappsecurity.com:80/bank/pay-bil	POST	name	{Post}name=Jason%2d%3e%3e	Existing	Existing	No	
http://zero.webappsecurity.com:80/bank/pay-bil	POST	account	{Post}name=Jason&address=1707	Existing	Existing	No	
http://zero.webappsecurity.com:80/bank/pay-bil	POST	address	{Post}name=Jason&address=1707	Existing	Existing	No	



Building Customer Success

Building Customer Success

- Cleared Professional Services
 - 1-Week Quick Starts
 - Long Term Staff Aug
- Multi-Day, Instructor Led On-Site Training Available

- On/Off Site Workshops, Brown Bags and Tech Sessions and Brown Bags (Free)
- Onsite short term PreSales consultation (Free)
- Customer Care Meetings (Free)
- On/Off Site User Group Meetings (Free)



Thank you !