Government Solutions

Securing DevOps, RMF and STIG

Fortify | WebInspect | Application Defender | Fortify-On-Demand

Defining Devops

State of Devops Report (Puppet, Dora): "..set of practices and cultural values that has been proven to help organizations of all sizes improve their software release cycles, software quality, **security**, and ability to get rapid feedback on product development. "

Amazon Web Services: "…cultural philosophies, practices, and tools that increases an organization's ability to **deliver applications and services at high velocity**"

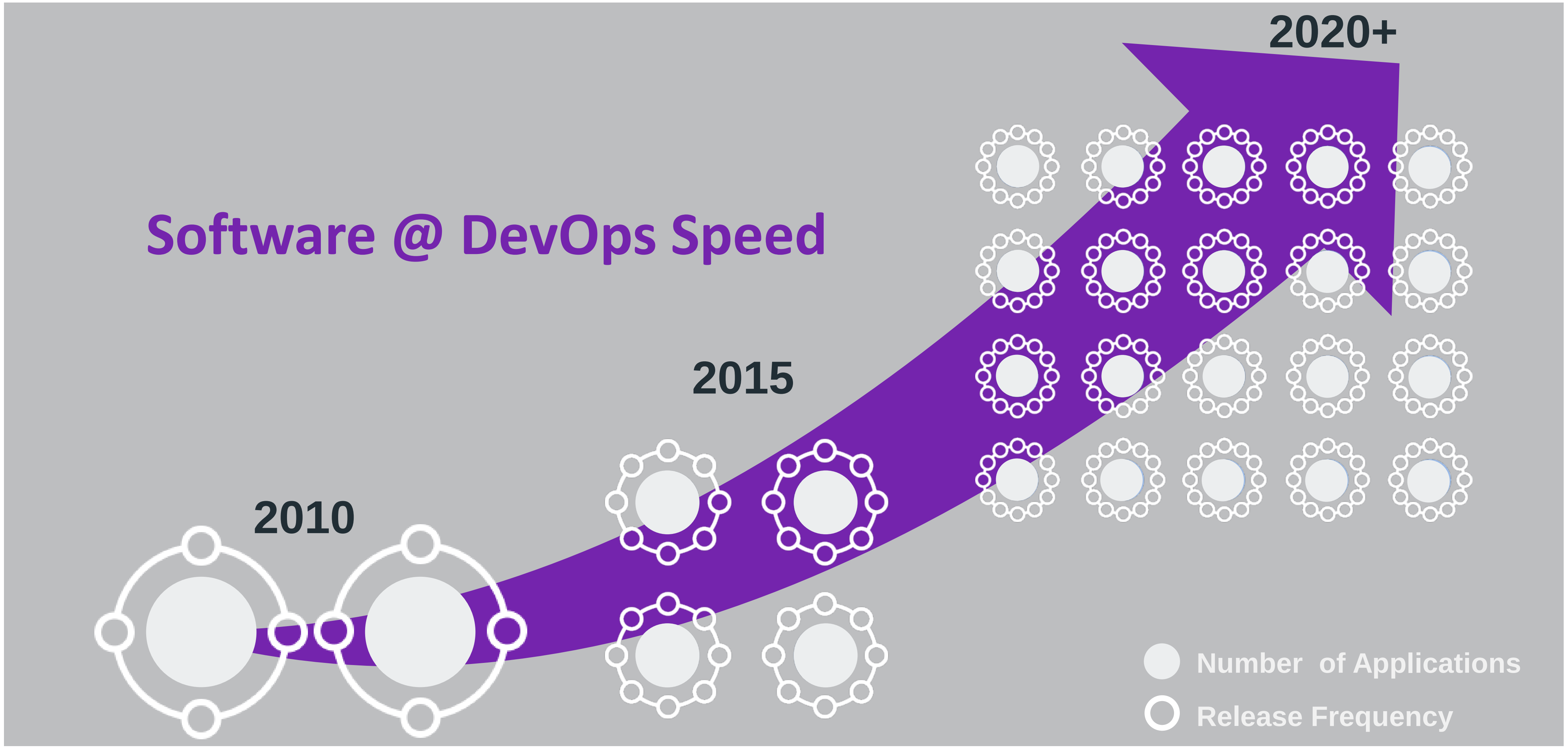
Wikipedia: **"**strongly advocate automation and monitoring at all steps of software construction, from integration, testing, releasing to deployment and infrastructure management. DevOps aims at **shorter development cycles**, increased deployment frequency, more dependable releases, in close alignment with business objectives.

Sources: 2017 State of DevOps Report, Presented by Puppet and DORA

Amazon Web Services : https://aws.amazon.com/devops/what-is-devops/

Wikipedia: https://en.wikipedia.org/wiki/DevOps

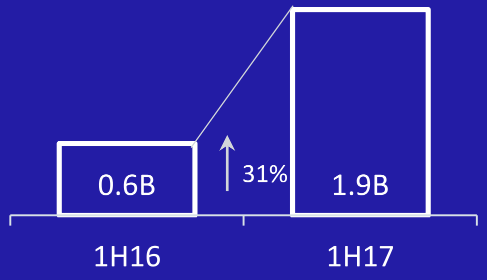
More Applications released 30x Faster



Source: Better outcomes, faster results. Continuous delivery and the race for better business performance,"
Forrester Thought Leader Paper commissioned by HP (now Hewlett Packard Enterprise)

AppSec Risk by the Numbers

1,900,000,000	Records lost globally in the first half of 2017
1,400,000	Sensitive PII records lost in a single US breach
15%	Survey respondents reported a breach
23%	Survey respondents citing their application as source

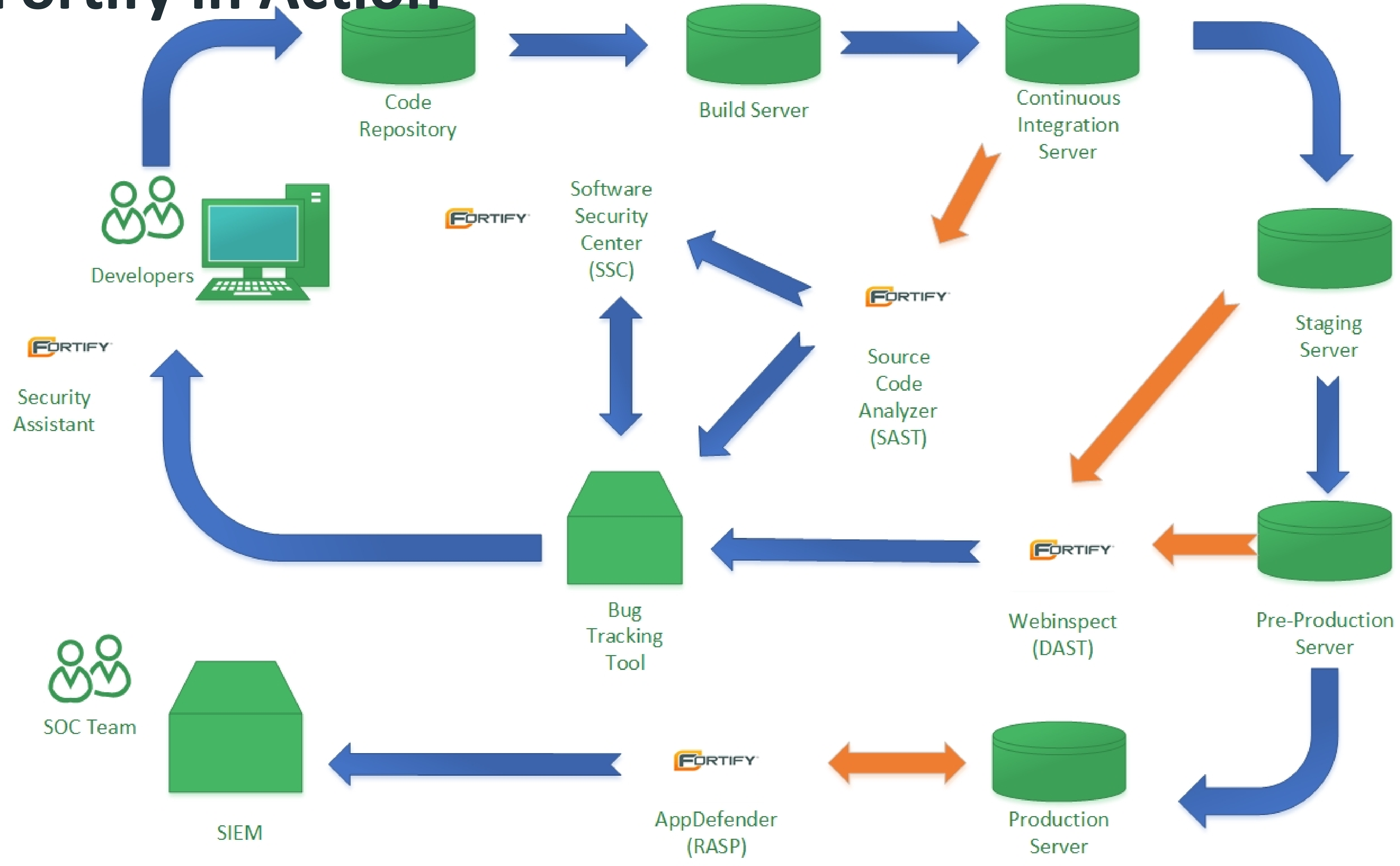


- References: breachlevelindex.com and SANS 2017 Application Security survey

“If there’s something we need to comply with, let’s turn it into an automated test.”

– Mark Schwartz, former CIO USCIS
DevOps Enterprise Summit 2014

Fortify in Action



Tenants of Dev Sec Ops

Automated Testing	Security Testing must be comprehensive and automated within the pipeline
Fail Fast	Able to fail the build based on security testing
Integrated Feedback	Immediate security feedback into into singular Issue Management
Fix Fast	Developers equipped to fix security issues rapidly
Cloud Deployable	Able to provision entire pipeline as code

Application Security Testing Challenges

Static Analysis

- Lengthy / Memory Intensive Scans for Large Applications
- Complex build processes can make integrations difficult for the security team
- Frequency of builds compounds the problem
- Large # of raw static findings that require human auditing to validate (this is #1)
- Organizational priority / risk tolerance needs to be applied to validated findings
- Managed service findings require prioritization as well
- Communication of findings to developers
- Communication of metrics / KPIs to management

Fortify Software Security Assurance

- Automated Comprehensive **Security Focused** static analysis
- Developers Plugins to take them to Vulnerable Line Of Code for fix, and Security Assistant for Prevention
- Build Adapters for automating build integration
- CI Plugins for scanning at build time and updating build status
- Issue Management Integrations
- Headless installs for cloud deployment into ephemeral environments

Best Programming Language Coverage

25+ Programming Languages supported and counting…

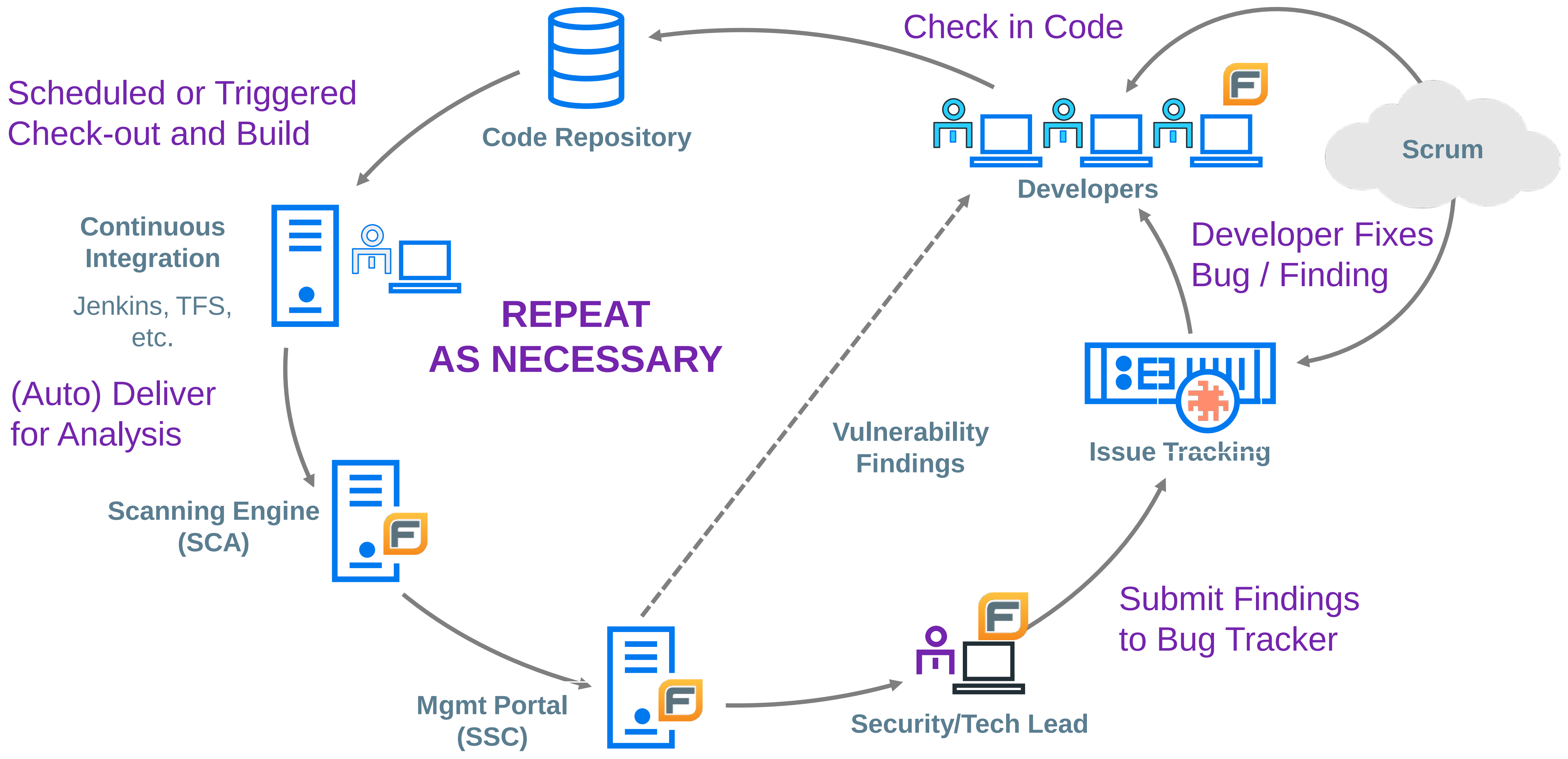
Java
Microsoft Visual C#
Microsoft Visual Basic
Swift
Scala
php
python

Ruby
JavaScript
HTML
<xml />
C
C++

Objective-C
salesforce Apex
SAP ABAP
COBOL
Microsoft Visual Basic 6.0
VBSCRIPT
ASP Active Server Pages

ORACLE PL SQL
T-SQL
AS
CF ADOBE COLDFUSION
JSP
Microsoft ASP.NET

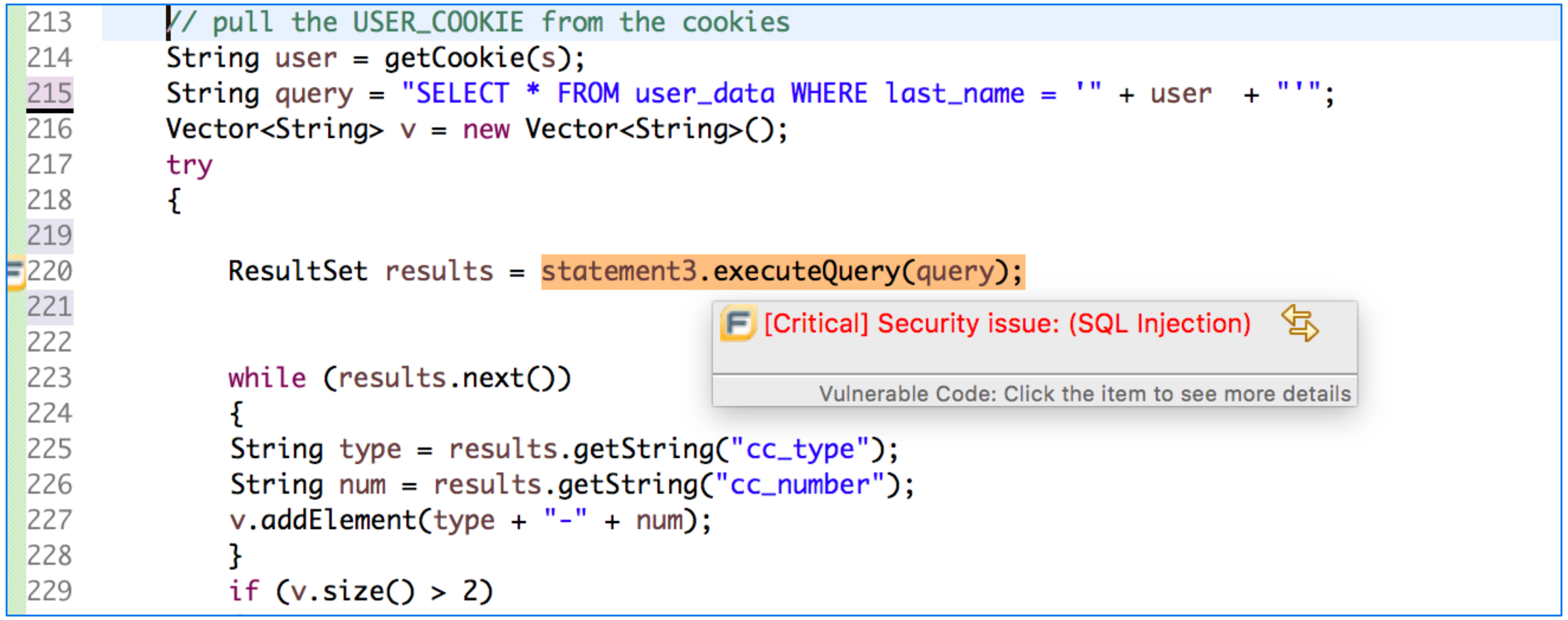
Static Software Scanning Process



MICRO FOCUS
Government Solutions

Developer Desktop – Security Assistant

Real time checking for most common issues *as you type*



```
// pull the USER_COOKIE from the cookies
String user = getCookie(s);
String query = "SELECT * FROM user_data WHERE last_name = '" + user  + "'";
Vector<String> v = new Vector<String>();
try
{

    ResultSet results = statement3.executeQuery(query);


    while (results.next())
    {
    String type = results.getString("cc_type");
    String num = results.getString("cc_number");
    v.addElement(type + "-" + num);
    }
    if (v.size() > 2)
```

[Critical] Security issue: (SQL Injection)

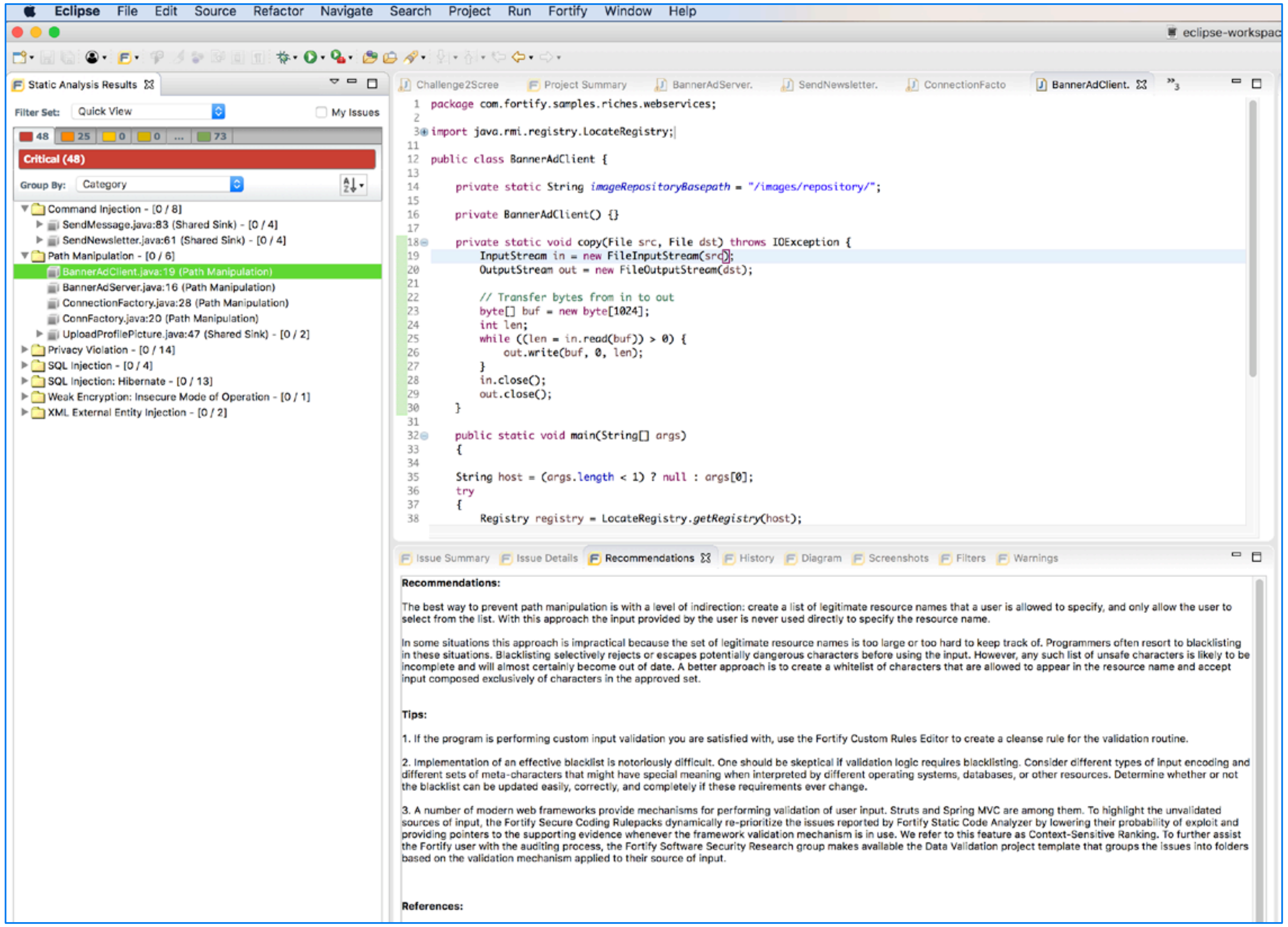
Vulnerable Code: Click the item to see more details

MICRO FOCUS Government Solutions

Developer Desktop – IDE Plugins

Scan and fix vulnerabilities before committing.

Open scan files generated from build integrations or security auditors for line of code detail – vulnerability overlaid on your code – and fix information



- Command Injection - [0 / 8]
  - SendMessage.java:83 (Shared Sink) - [0 / 4]
  - SendNewsletter.java:61 (Shared Sink) - [0 / 4]
- Path Manipulation - [0 / 6]
  - BannerAdClient.java:19 (Path Manipulation)
  - BannerAdServer.java:16 (Path Manipulation)
  - ConnectionFactory.java:28 (Path Manipulation)
  - ConnFactory.java:20 (Path Manipulation)
  - UploadProfilePicture.java:47 (Shared Sink) - [0 / 2]
- Privacy Violation - [0 / 14]
- SQL Injection - [0 / 4]
- SQL Injection: Hibernate - [0 / 13]
- Weak Encryption: Insecure Mode of Operation - [0 / 1]
- XML External Entity Injection - [0 / 2]

```
package com.fortify.samples.riches.webservices;

import java.rmi.registry.LocateRegistry;

public class BannerAdClient {

    private static String imageRepositoryBasepath = "/images/repository/";

    private BannerAdClient() {}

    private static void copy(File src, File dst) throws IOException {
        InputStream in = new FileInputStream(src);
        OutputStream out = new FileOutputStream(dst);

        // Transfer bytes from in to out
        byte[] buf = new byte[1024];
        int len;
        while ((len = in.read(buf)) > 0) {
            out.write(buf, 0, len);
        }
        in.close();
        out.close();
    }

    public static void main(String[] args)
    {

    String host = (args.length < 1) ? null : args[0];
    try
    {
        Registry registry = LocateRegistry.getRegistry(host);
```

Recommendations:

The best way to prevent path manipulation is with a level of indirection: create a list of legitimate resource names that a user is allowed to specify, and only allow the user to select from the list. With this approach the input provided by the user is never used directly to specify the resource name.

In some situations this approach is impractical because the set of legitimate resource names is too large or too hard to keep track of. Programmers often resort to blacklisting in these situations. Blacklisting selectively rejects or escapes potentially dangerous characters before using the input. However, any such list of unsafe characters is likely to be incomplete and will almost certainly become out of date. A better approach is to create a whitelist of characters that are allowed to appear in the resource name and accept input composed exclusively of characters in the approved set.

Tips:

1. If the program is performing custom input validation you are satisfied with, use the Fortify Custom Rules Editor to create a cleanse rule for the validation routine.

2. Implementation of an effective blacklist is notoriously difficult. One should be skeptical if validation logic requires blacklisting. Consider different types of input encoding and different sets of meta-characters that might have special meaning when interpreted by different operating systems, databases, or other resources. Determine whether or not the blacklist can be updated easily, correctly, and completely if these requirements ever change.

3. A number of modern web frameworks provide mechanisms for performing validation of user input. Struts and Spring MVC are among them. To highlight the unvalidated sources of input, the Fortify Secure Coding Rulepacks dynamically re-prioritize the issues reported by Fortify Static Code Analyzer by lowering their probability of exploit and providing pointers to the supporting evidence whenever the framework validation mechanism is in use. We refer to this feature as Context-Sensitive Ranking. To further assist the Fortify user with the auditing process, the Fortify Software Security Research group makes available the Data Validation project template that groups the issues into folders based on the validation mechanism applied to their source of input.

References:

Build Integrations with Fortify Source Code Analyzer

Build Integrations make it easy to integrate automated static analysis into the complete build process.
Out of the box support for a wide variety of BUILD TOOLS
Robust Fortify Command Line utilities exist for additional integrations.

Ant 1.9.6

Gradle 2.13 The Fortify Static Code Analyzer Gradle build integration supports the following language/platform combinations:

- Java/Windows, Linux, and macOS
- C/Linux
- C++/Linux

Jenkins 1.6

Maven 3.0.5, 3.3.x

MSBuild 4.x, 12.0,14.0, 15.0

Xcodebuild 8.0, 8.1, 8.2, 8.3

Build Integrations Out of the Box

GNU Make

Maven™

Gradle

MSBuild

<APACHE ANT>

XCodeBuild

Build Integrations make it easy to integrate automated static analysis into the complete build process.
Out of the box support for a wide variety of BUILD TOOLS
Robust Fortify Command Line utilities exist for additional integrations.

Continuous Integration Plugins

Bamboo

Team Foundation Server

Visual Studio Team Services

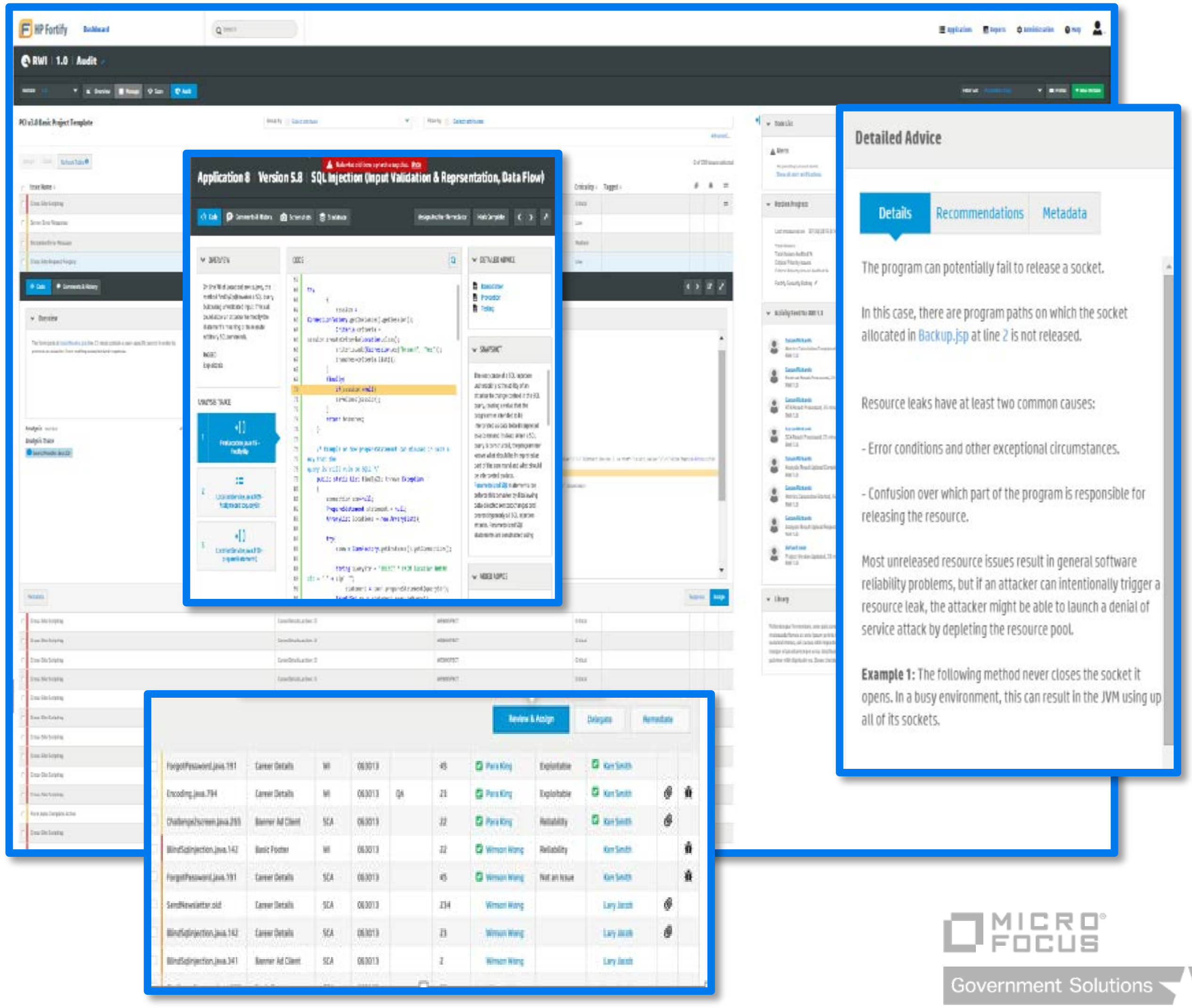
Fortify Client

Fortify FPR Utility

Software Security Center

Enterprise Ready Software Security Management

- Artifact and Vulnerability Management
- Portfolio level KPI's and Metrics
- Open Reporting Interface
- Swaggerized RESTFul APIs
- LDAP/SSO/CAC Ready



Detailed Advice

The program can potentially fail to release a socket.

In this case, there are program paths on which the socket allocated in Backup.jsp at line 2 is not released.

Resource leaks have at least two common causes:

- Error conditions and other exceptional circumstances.

- Confusion over which part of the program is responsible for releasing the resource.

Most unreleased resource issues result in general software reliability problems, but if an attacker can intentionally trigger a resource leak, the attacker might be able to launch a denial of service attack by depleting the resource pool.

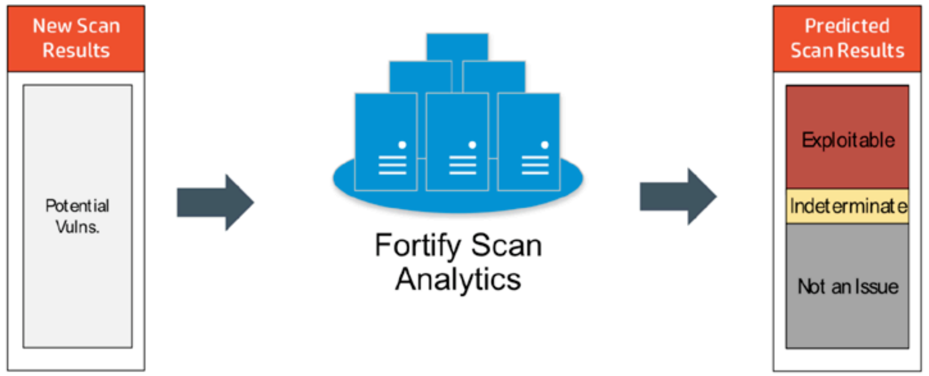
Example 1: The following method never closes the socket it opens. In a busy environment, this can result in the JVM using up all of its sockets.

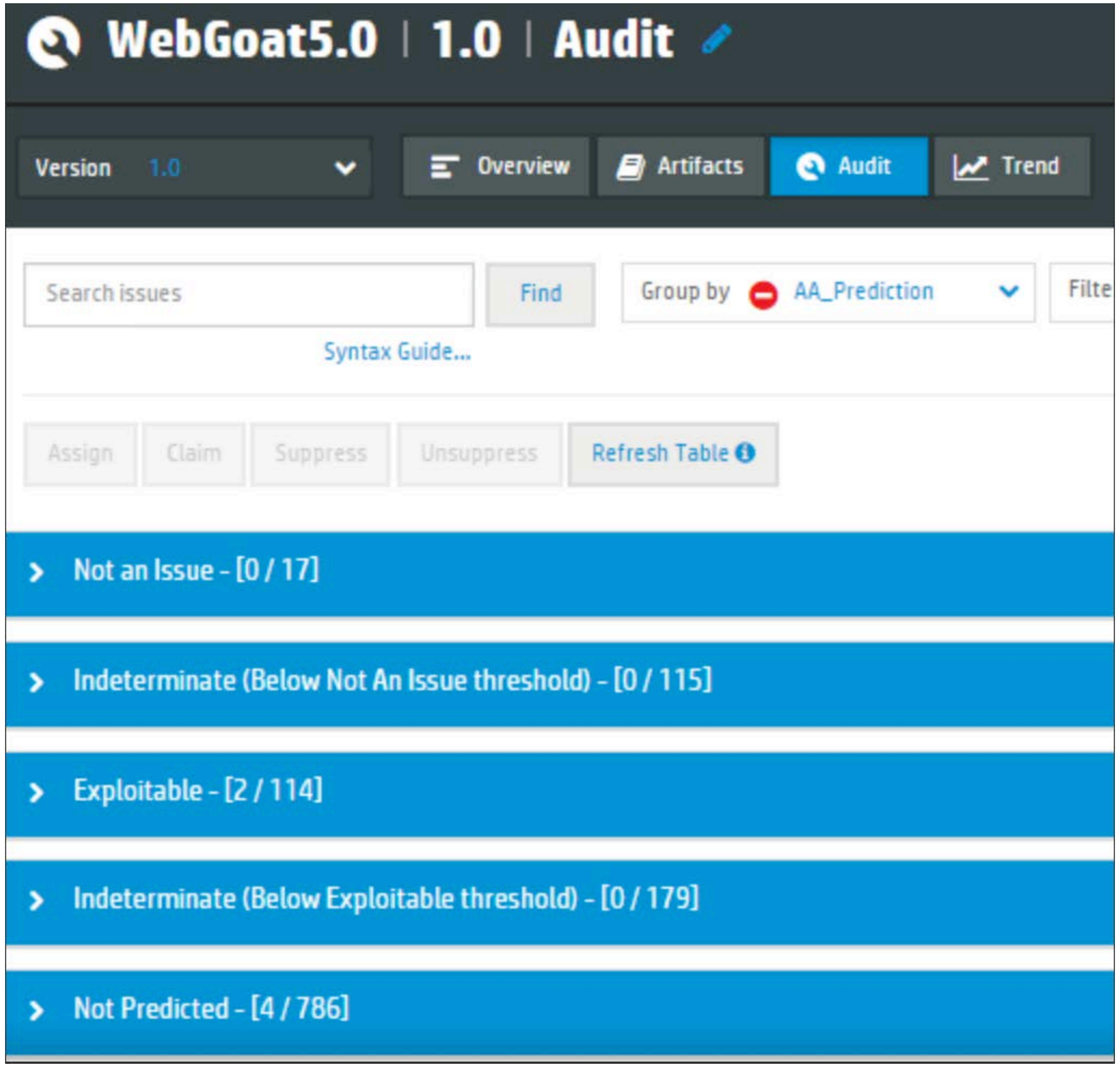
ForgotPassword.java.191	Career Details	WI	08.2013		45	✓ Paris King	Exploitable	✓ Ken Smith		
Encoding.java.794	Career Details	WI	08.2013	QA	23	✓ Paris King	Exploitable	✓ Ken Smith	📎	👤
ChallengeScreen.java.293	Banner Ad Client	SCA	08.2013		22	✓ Paris King	Reliability	✓ Ken Smith	📎	
BlindSqlInjection.java.142	Basic Footer	WI	08.2013		22	✓ Winson Wong	Reliability	Ken Smith		👤
ForgotPassword.java.191	Career Details	SCA	08.2013		45	✓ Winson Wong	Not an Issue	Ken Smith		👤
SendNewsletter.old	Career Details	SCA	08.2013		234	Winson Wong		Lary Jacob	📎	
BlindSqlInjection.java.142	Career Details	SCA	08.2013		23	Winson Wong		Lary Jacob	📎	
BlindSqlInjection.java.341	Banner Ad Client	SCA	08.2013		2	Winson Wong		Lary Jacob		

MICRO FOCUS
Government Solutions

Audit Assistant

- Machine learning to make AppSec more efficient
- Identify true vulnerabilities with up to 98% accuracy and prioritize them for remediation faster
- Focus on triaging and investigating high priority vulnerabilities.
- Return value-added time to your developers and auditors





Not an Issue - [0 / 17]

Indeterminate (Below Not An Issue threshold) - [0 / 115]

Exploitable - [2 / 114]

Indeterminate (Below Exploitable threshold) - [0 / 179]

Not Predicted - [4 / 786]

Issue Management Integrations

Bugzilla

MICRO FOCUS® Application Lifecycle Management

JIRA

Fortify Service Integration Extendable Plugin Architecture

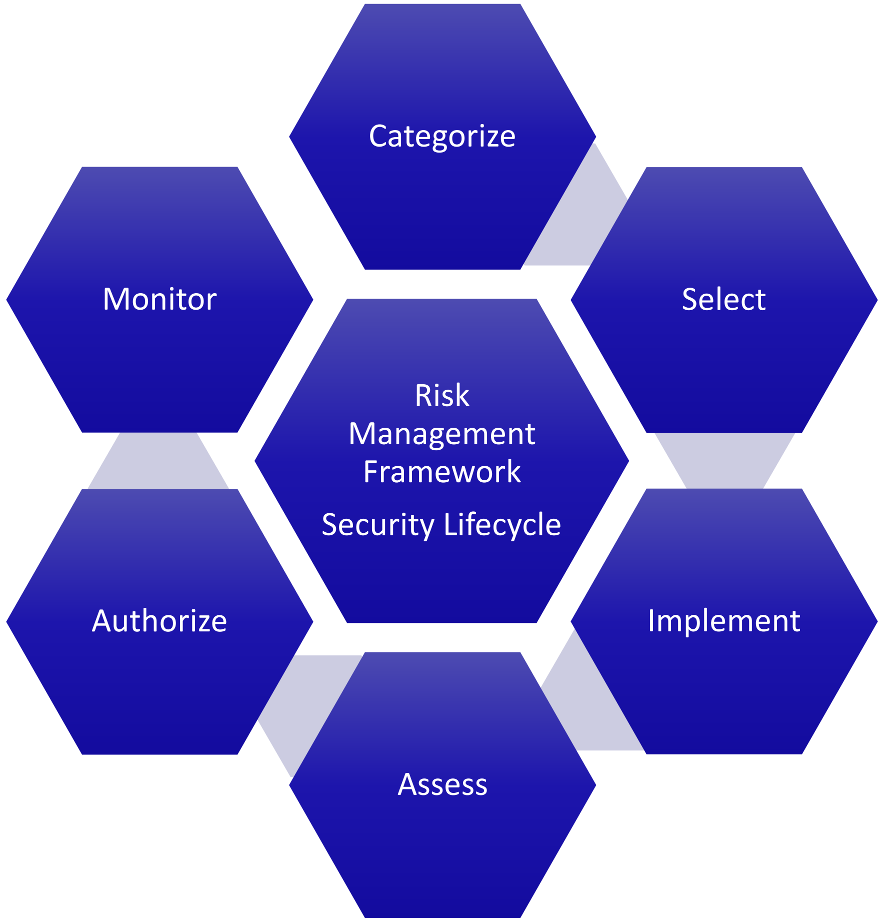
Visual Studio Team Services

Team Foundation Server

RMF and Fortify

What is RMF?

- The Risk Management Framework (RMF) for DoD Information Technology (IT) (DoDI 8510.01)
  - “Formalizes set of standards and used by DoD agencies to ensure that the security posture of a given system is acceptable and is maintained throughout it’s lifecycle.”
- 6 Step approach used for the Authorization of Federal IT Systems.



Type of tests

- Controls Assessment
  - NIST 800-53A
- STIG/SRG/DON/USMC Policy
  - Manual
  - Benchmark
    - SCAP
- Vulnerability Scans

Application Security compliance requirements change

DISA Application Security and Development STIG V5

AppStig provides “principles and guidelines” for with DoD cybersecurity policies, standards, architectures, security controls, and validation procedures. New in 4.x is mapping of Stig controls to NIST 800-53 rev4 controls through control correlation identifiers (CCI)

Increasing requirements

The number of controls has gone from 158 to 290.

Includes both quality and security issues

Required Validation

Dynamic: APSC-DV-001460 **CAT II**, titled “An application vulnerability assessment **must** be conducted.”

Static: APSC-DV-003170 **CAT II**, titled “An application code review **must** be performed on the application.”

Fortify for RMF/STIG

- RMF refers to NIST’s categorizations
- STIG checks form the bulk of the compliance testing that will be done as part of the RMF process.
- Accounts for >50% of the testing involved in a typical system.
- Application STIG is mapped to NIST’s categorizations through Control Correlation Identifier (CCI)
- Fortify (SCA, SSC, WebInspect and Application Defender) map directly to NIST 800-53R4 and STIG 4.x

Automation is your new best friend

- New Requirements
  - Additional controls in the new STIG
  - Need to map to CCI
  - Generate and Manage POA&Ms
- Solutions
  - Automate build and scan process
  - Use Audit Assistant, Application Defender and other innovative technologies.
  - Use APIs to automate processes
  - Use specific additional tools that have integrated with Fortify to generate and manage POA&Ms
    - PAGE tool – Developed for Navy/USMC use

PAGE Overview

Plan Of Action & Milestone (POA&M) Automated Generation Engine

Description

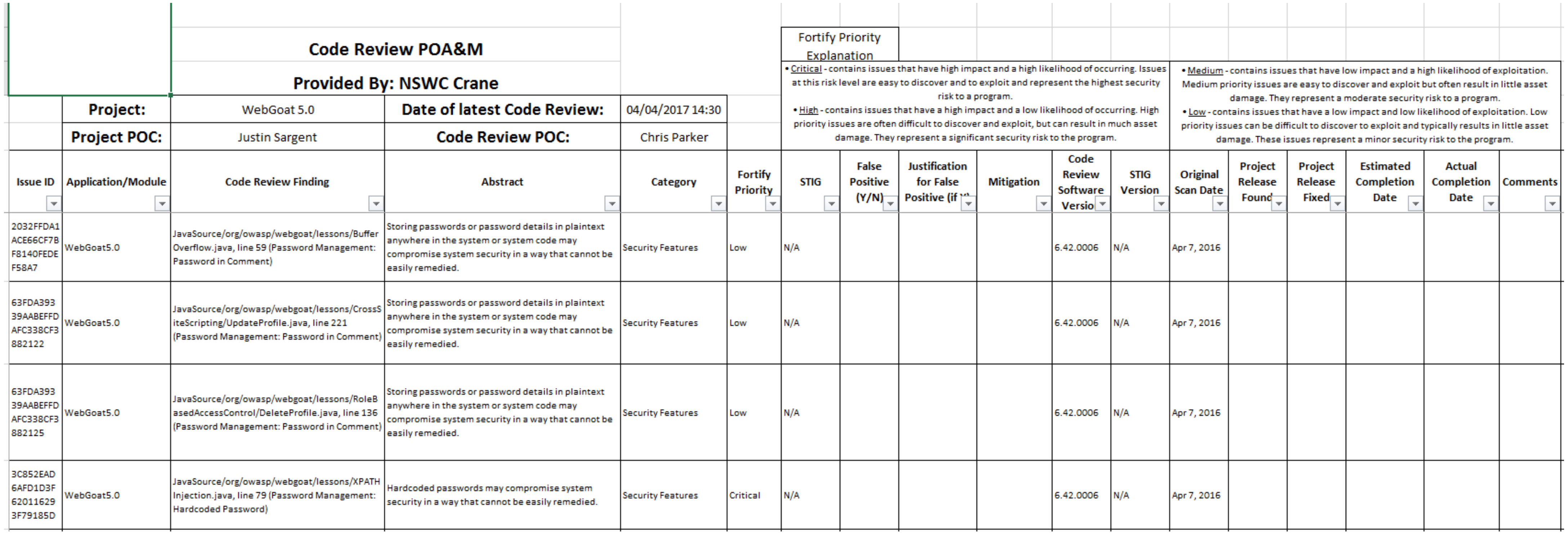
- Maintained by NSWC Crane Tactical Cyber Innovation Team (TaCIT)
- Supports: ACAS, SCAP, STIG and Fortify
- Provides detailed information needed for remediation

Capabilities

- Directly transform scan results files into POA&M documents in DoD standard format
- See results info all in one place
- Keep living documents, with in-place updates
- Reduce turnaround time for POA&M documents

Sample PAGE Output

Microsoft Excel file



Code Review POA&M

Provided By: NSWC Crane

Project:	WebGoat 5.0	**Date of latest Code Review:**	04/04/2017 14:30
Project POC:	Justin Sargent	**Code Review POC:**	Chris Parker

Fortify Priority Explanation

• Critical - contains issues that have high impact and a high likelihood of occurring. Issues at this risk level are easy to discover and to exploit and represent the highest security risk to a program.
• High - contains issues that have a high impact and a low likelihood of occurring. High priority issues are often difficult to discover and exploit, but can result in much asset damage. They represent a significant security risk to the program.

• Medium - contains issues that have low impact and a high likelihood of exploitation. Medium priority issues are easy to discover and exploit but often result in little asset damage. They represent a moderate security risk to a program.
• Low - contains issues that have a low impact and low likelihood of exploitation. Low priority issues can be difficult to discover to exploit and typically results in little asset damage. These issues represent a minor security risk to the program.

Issue ID	Application/Module	Code Review Finding	Abstract	Category	Fortify Priority	STIG	False Positive (Y/N)	Justification for False Positive (if	Mitigation	Code Review Software Versio	STIG Version	Original Scan Date	Project Release Found	Project Release Fixed	Estimated Completion Date	Actual Completion Date	Comments
2032FFDA1ACE66CF7BF8140FEDEF58A7	WebGoat5.0	JavaSource/org/owasp/webgoat/lessons/BufferOverflow.java, line 59 (Password Management: Password in Comment)	Storing passwords or password details in plaintext anywhere in the system or system code may compromise system security in a way that cannot be easily remedied.	Security Features	Low	N/A				6.42.0006	N/A	Apr 7, 2016					
63FDA39339AABEFFDAFC338CF3882122	WebGoat5.0	JavaSource/org/owasp/webgoat/lessons/CrossSiteScripting/UpdateProfile.java, line 221 (Password Management: Password in Comment)	Storing passwords or password details in plaintext anywhere in the system or system code may compromise system security in a way that cannot be easily remedied.	Security Features	Low	N/A				6.42.0006	N/A	Apr 7, 2016					
63FDA39339AABEFFDAFC338CF3882125	WebGoat5.0	JavaSource/org/owasp/webgoat/lessons/RoleBasedAccessControl/DeleteProfile.java, line 136 (Password Management: Password in Comment)	Storing passwords or password details in plaintext anywhere in the system or system code may compromise system security in a way that cannot be easily remedied.	Security Features	Low	N/A				6.42.0006	N/A	Apr 7, 2016					
3C852EAD6AFD1D3F620116293F79185D	WebGoat5.0	JavaSource/org/owasp/webgoat/lessons/XPATHInjection.java, line 79 (Password Management: Hardcoded Password)	Hardcoded passwords may compromise system security in a way that cannot be easily remedied.	Security Features	Critical	N/A				6.42.0006	N/A	Apr 7, 2016					

PAGE Turnaround

- Manually creating POA&Ms with 5K findings took, on average, ~**40 hours**
  - Human error, bad copy/paste, etc.

- Creating POA&Ms with PAGE with 5k findings takes, on average, ~**2 minutes**
  - Proper formatting, findings de-duplicated

Thank you

- Remove barriers to implementing security in software application development by better integration
- Use tools designed for STIG and RMF purposes

The agility you need with the results you want

MICRO FOCUS
Government Solutions