May 2017

Hewlett Packard Enterprise

Capabilities to help expand and mature SWA program

Haleh Nematollahy – Sr. Security Solutions Architect

Fortify Security Assistant

Fortify security assistant

Building in security as you code

Identify weaknesses as developers write code in real-time

Spell check security scanning

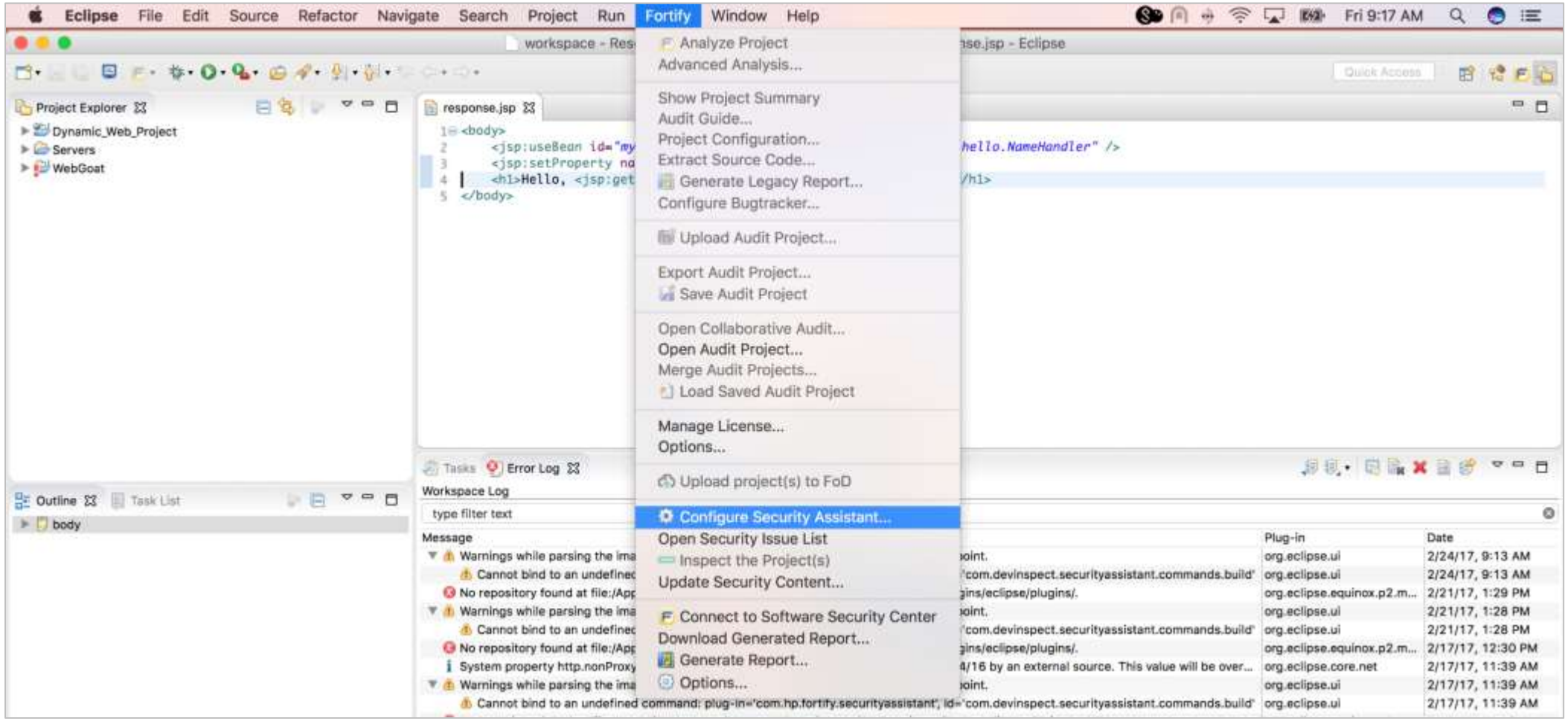
Identify issues earlier in the SDLC

Educate developer about security

Accelerate appsec program (increase productivity & efficiency)

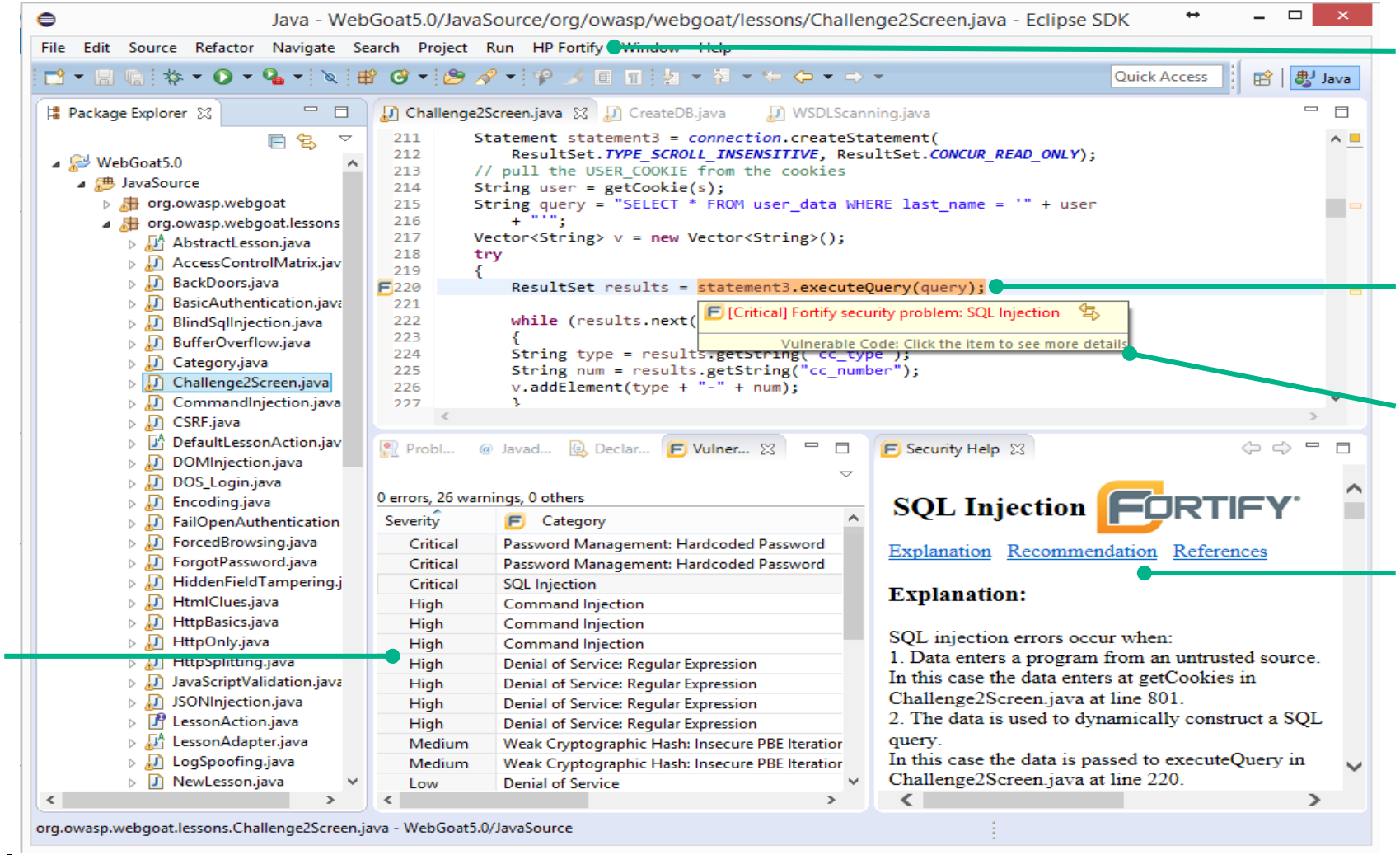
Hewlett Packard Enterprise

Fortify security assistant



Fortify security assistant

Real-time lightweight analysis of the source code



Fortify menu for additional options

Vulnerable line of code is highlighted as developer code & provides tips for additional information

Level of criticality

Type of vulnerability, explanation and detailed remediation guidance

All issues detected in the project

Hewlett Packard Enterprise

Audit Assistant/ Scan Analytics

Machine Learning - scan analytics & audit assistant

Do more with your AppSec DATA

Streamline appsec program by making the auditing process more efficient

Increase the relevancy and consistency of findings unique to your organization preferences

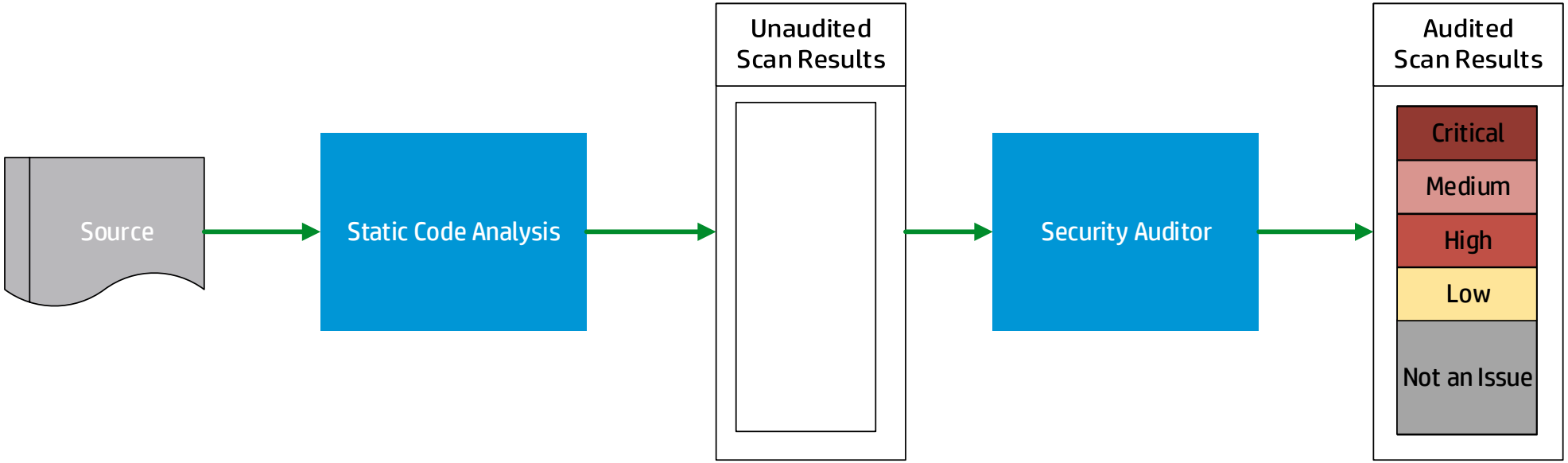
Identify relevant issues earlier in the SDLC

Scale and accelerate your AppSec program with existing resources

Hewlett Packard Enterprise

Static analysis workflow

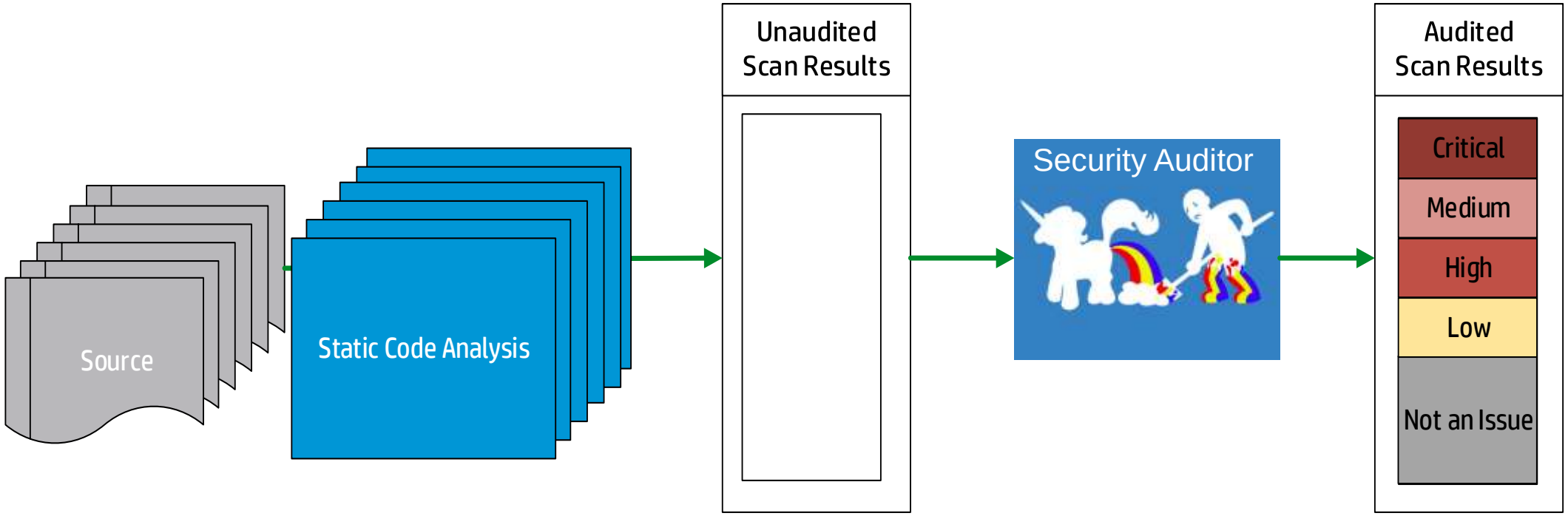
Or: How I scan a singe application



Finding relevant scan results is expensive and hard to scale because it requires:

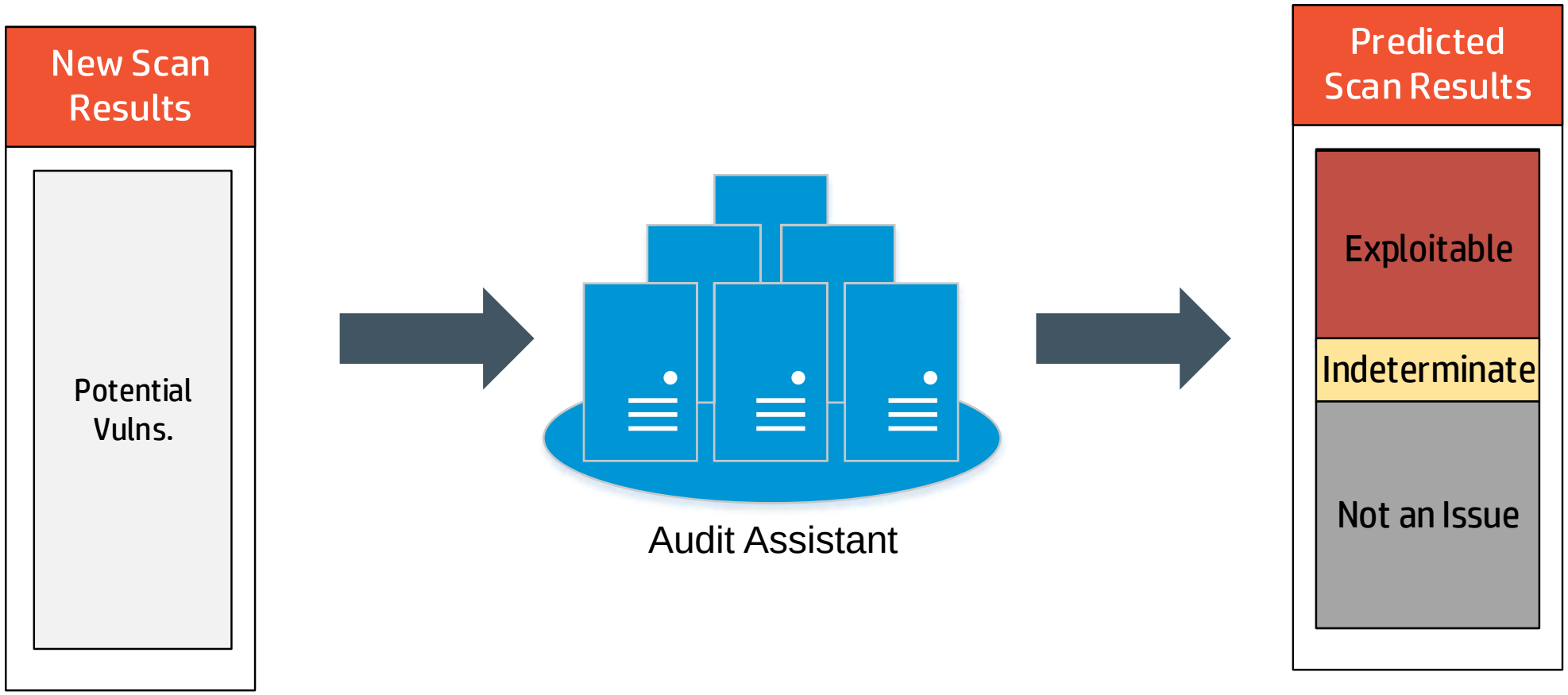
- Security expertise
- Knowledge of scanned application's context

Challenge: Identifying Issues at scale can be painstaking



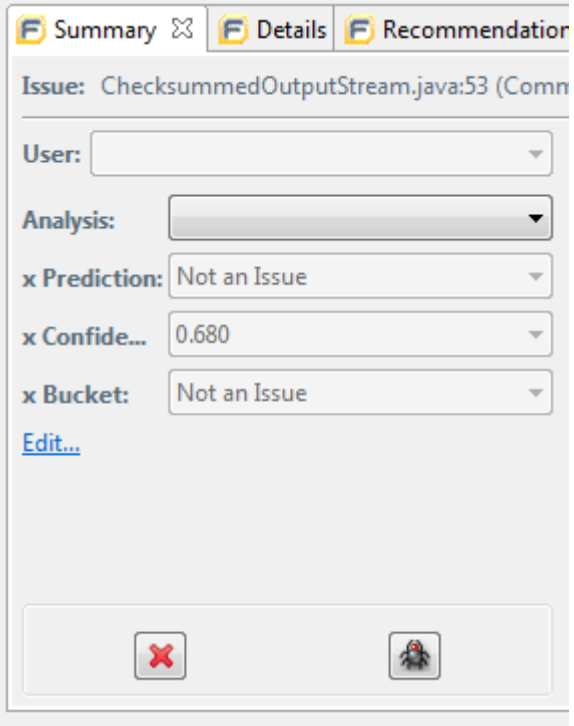
Software Security Center (SSC) - Audit assistant

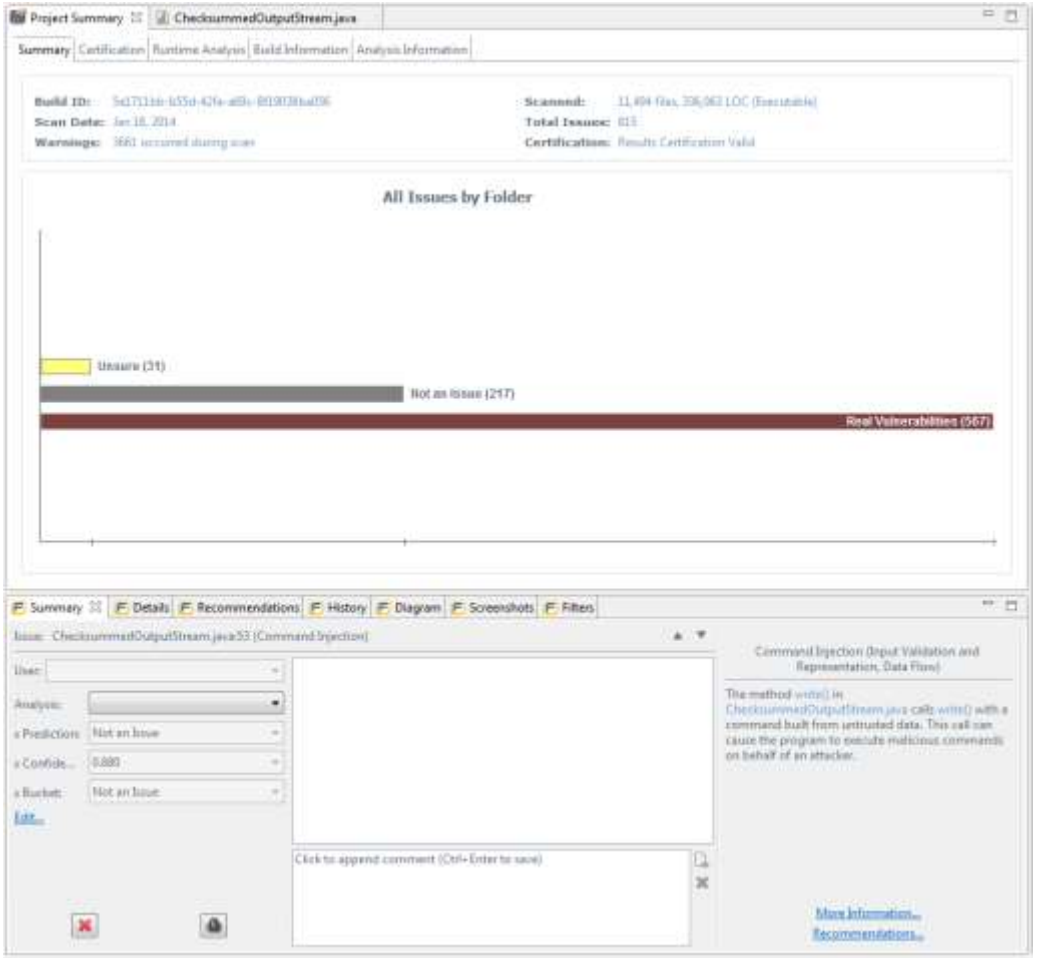
Machine learning assisted identification of relevant scan results



Seamless workflow integration with existing tools

Audit Workbench – Security Auditor's View



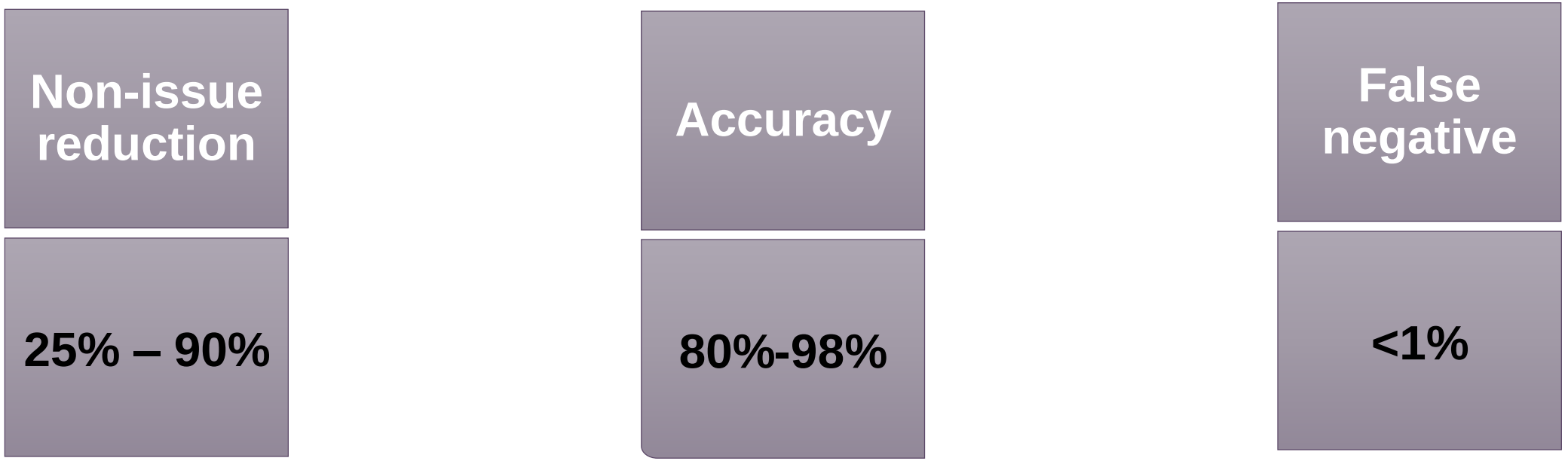


All product views are illustrations and might not represent actual product screens

Hewlett Packard Enterprise

Return value-added time to auditors and developers

Without sacrificing scan integrity

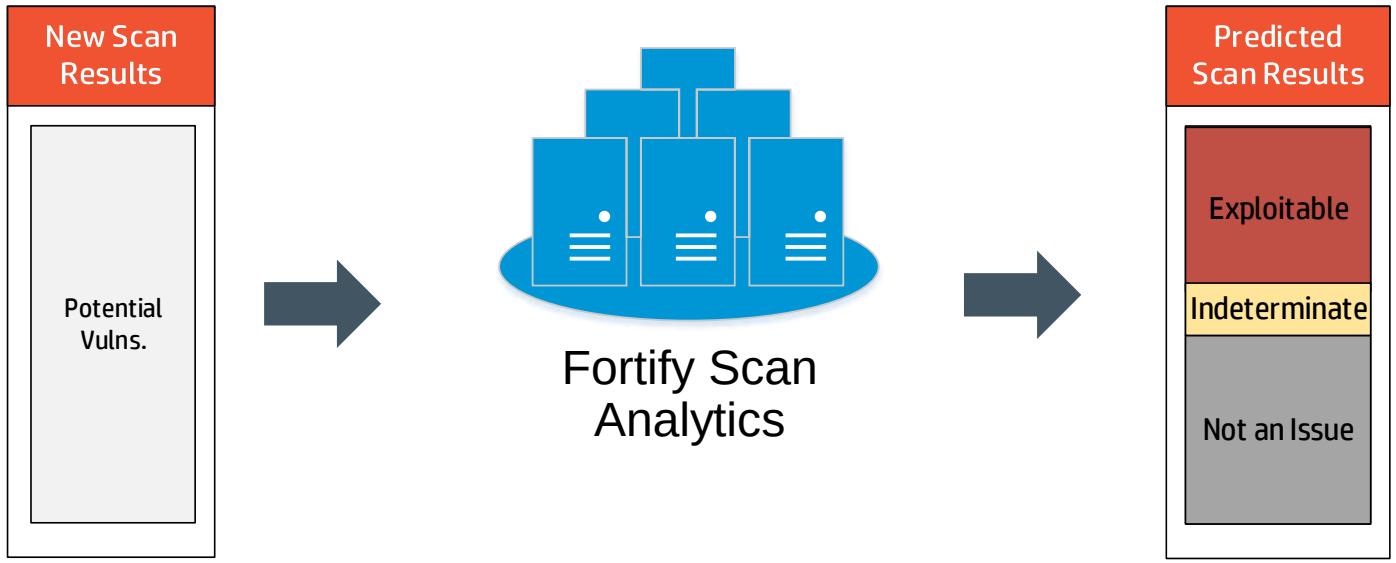


Results obtained are based on real world applications and scenarios.
Results vary based on training and customization. They are not guarantees of future performance.

Scan analytics

Machine learning to make AppSec more efficient

- Identify true vulnerabilities and prioritize them for remediation faster
- Focus on triaging and investigating high priority vulnerabilities.
- Return value-added time to your developers and auditors



New and Improved .Net scanning

Scanning with our new .NET front end

Before 16.20	After 16.20
Pre-compilation required	No pre-compilation
Compilation required	No compilation necessary
DLL only translation	Source code translation
Visual Studio required - Compilation	Visual Studio required BUT for completely different reasons - Command-line generation

.NET translation options

– vsversion option is obsolete

.NET Command-Line Options

The following table describes the .NET command-line options.

Note: These options are not required if you translate the code with the Visual Studio Command Prompt and you have HPE Security Fortify Package for Visual Studio installed.

.NET Option	Description
`-dotnet-version <version>`	Specifies the .NET framework version. See the *HPE Security Fortify Software System Requirements* for a list of supported versions. This adds the location of .NET framework libraries (DLLs) for the specified .NET framework version to the list of directories/paths specified by the `-libdirs` option, unless the `-libdirs-only` option is specified.
`-libdirs <dirs>|<paths>`	Specifies a semicolon-separated list of directories where referenced system or third-party DLLs are located. You can also specify paths to specific DLLs with this option.
`-libdirs-only`	Sets the list of directories or paths to only those specified by the `-libdirs` option. Otherwise, Fortify Static Code Analyzer includes the location of the .NET framework libraries (DLLs) that correspond to the .NET framework version specified with the `-dotnet-version` option.
`-dotnet-preproc-symbols <symbols>`	Specifies a semicolon-separated list of preprocessor symbols used in the source code. For example: `-dotnet-preproc-symbols "DEBUG;TRACE"`
`-dotnet-assembly-name <assembly_name>`	Specifies the name of the target .NET assembly as specified in Visual Studio project settings.
`-dotnetwebroot <root_dir>`	**.NET Web projects only.** Specifies the home directory of an ASP.NET project.
`-cs-extern-alias <aliases_path_pairs>`	**C# projects only.** Specifies a list of external aliases for a specified DLL file in the following format: `alias1,alias2,..=<path_to_DLL>`. If multiple DLLs are assigned external aliases, specify multiple `-cs-extern-alias` options on the command line.
`-vb-root <namespace>`	**.VB.NET projects only.** Specifies the root namespace for the project as specified in Visual Studio project settings.
`-vb-imports <namespaces>`	**VB.NET projects only.** Specifies a semicolon-separated list of namespaces imported for all source files in the project.

.NET Option	Description
`-vb-mytype <symbol>`	**VB.NET projects only.** Specifies the value for the `_MYTYPE` preprocessor symbol that is specified in the `<MyType>` tag in the project settings. This is required if the source code to be translated uses My namespace.
`-vb-webproject`	**VB.NET projects only.** Indicates that the project is a pure Web project (no code-behind the source files).
`-vb-compile-options <compile_options>`	**VB.NET projects only.** Specifies any special compilation options required for the correct translation of the source code, such as `OptionStrict`, `OptionInfer`, and `OptionExplicit`. The format for *<compile_options>* is a comma-separated list of: *<option>*`=On|Off`. For example: `-vb-compile-options "OptionStrict=On,OptionExplicit=Off"`
`-vsversion <version>`	(Deprecated - Replaced by `-dotnet-version` option) Specifies the version number that corresponds to your Visual Studio version. • Visual Studio 2012: `11.0` • Visual Studio 2013: `12.0` • Visual Studio 2015: `14.0`

Hewlett Packard Enterprise

Sample Translation File (Generated by 17.10 VS plugin)

"-b“ "WebGoat.NET"

"-machine-output"

"-dotnet-assembly-name“ "DotNetGoat"

"-cs-extern-alias "global="- <.Net framework DLLs, User lib DLLs>“

-dotnet-preproc-symbols“ "DEBUG"

"-dotnetwebroot“ "C:\Users\nematoll\Documents\nematollVS2017\OWASP-WebGoat.NET-e9603b9\WebGoat\\"

"-libdirs-only"

"-libdirs“ "<.Net framework DLLs, User lib DLLs>"

"-dotnet-applibs“ "<User bin DLLs>""

"C:\VS2017\OWASP-WebGoat.NET-e9603b9\WebGoat\AddNewUser.aspx"

"C:\VS2017\OWASP-WebGoat.NET-e9603b9\WebGoat\AddNewUser.aspx.cs"

"C:\VS2017\OWASP-WebGoat.NET-e9603b9\WebGoat\AddNewUser.aspx.designer.cs"

<… All source files – C#, VB.NET, ASPX, ASCX, XML, CONFIG etc.>

Key Points

Scanning: Ease of use	Scanning is a lot more simple. No need for pre-compilation is a game changer from ease of use perspective
Scanner Architecture: Build vs Folder	You may do folder scans now but leverage Visual Studio to build the translate command until you feel confident
Results: WebForms Application (ASPX)	Significantly improved since 16.2

Extended .NET Frontend supports async/await

- Async/await support
  - SCA can now find vulnerabilities in aync/await constructs
- New file extension support
  - SCA can now scan .winmd files
- SCA can now scan Silverlight apps
- SCA supports latest versions of:
  - C# 6
  - VB.Net 14
- MSbuild support improvements

Parallel Processing/ Multithreaded scanning

Parallel scanning – The old way

– Introduced in version 4.0

– SCA used to spawn multiple processes in parallel

– Process was resource heavy and required mathematical computation

  – For example, if the machine had 32 GB of RAM and 8 cores, the recommended configuration would be:

    sourceanalyzer … -j 2 –Xmx14G –Dcom.fortify.sca.RmiWorkerMaxHeap=7G

    … And this may still lead to memory errors depending on complexity of code.

Deprecated as of 17.10

Multithreaded parallel mode

Solution

– Redesigned and reimplemented
  – Uses native Java multithreading instead of creating master process and spawning separate processes
  – Removed need for communications and monitoring between master and child processes – added burden
– Simple to enable (no complex mathematics)

Results

– Scans complete in 50% of the time compared to single-threaded on average
– Process is optimized and scales to available resources automatically

Risks

– Running the SCA security analyzers in multithreaded mode may introduce non-determinism into issue results. If this occurs, switch back to single-threaded scan.