Hewlett Packard Enterprise

Securing DevOps, RMF and STIG

Scott Snowden
Sameer Kamani

May 2017 – San Diego Federal Fortify Users Group

DevOps – definition and principles

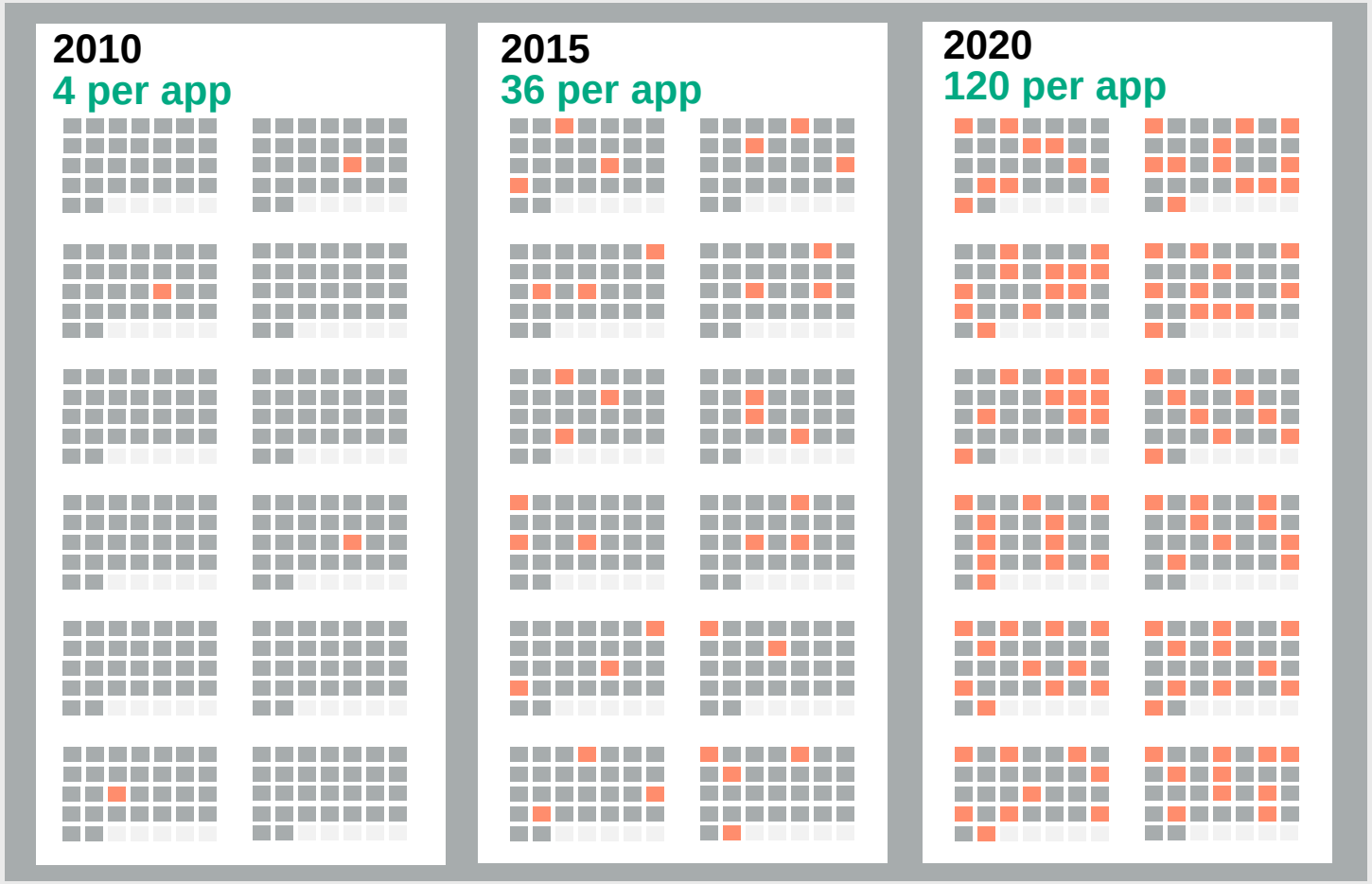
DevOps (a clipped compound of development and operations) is a practice that emphasizes the collaboration and communication of both software developers and other information-technology (IT) professionals while automating the process of software delivery and infrastructure changes.

Principles

- Develop and test in an environment similar to production
- Deploy builds frequently
- Validate quality continuously

Shorter release cycles

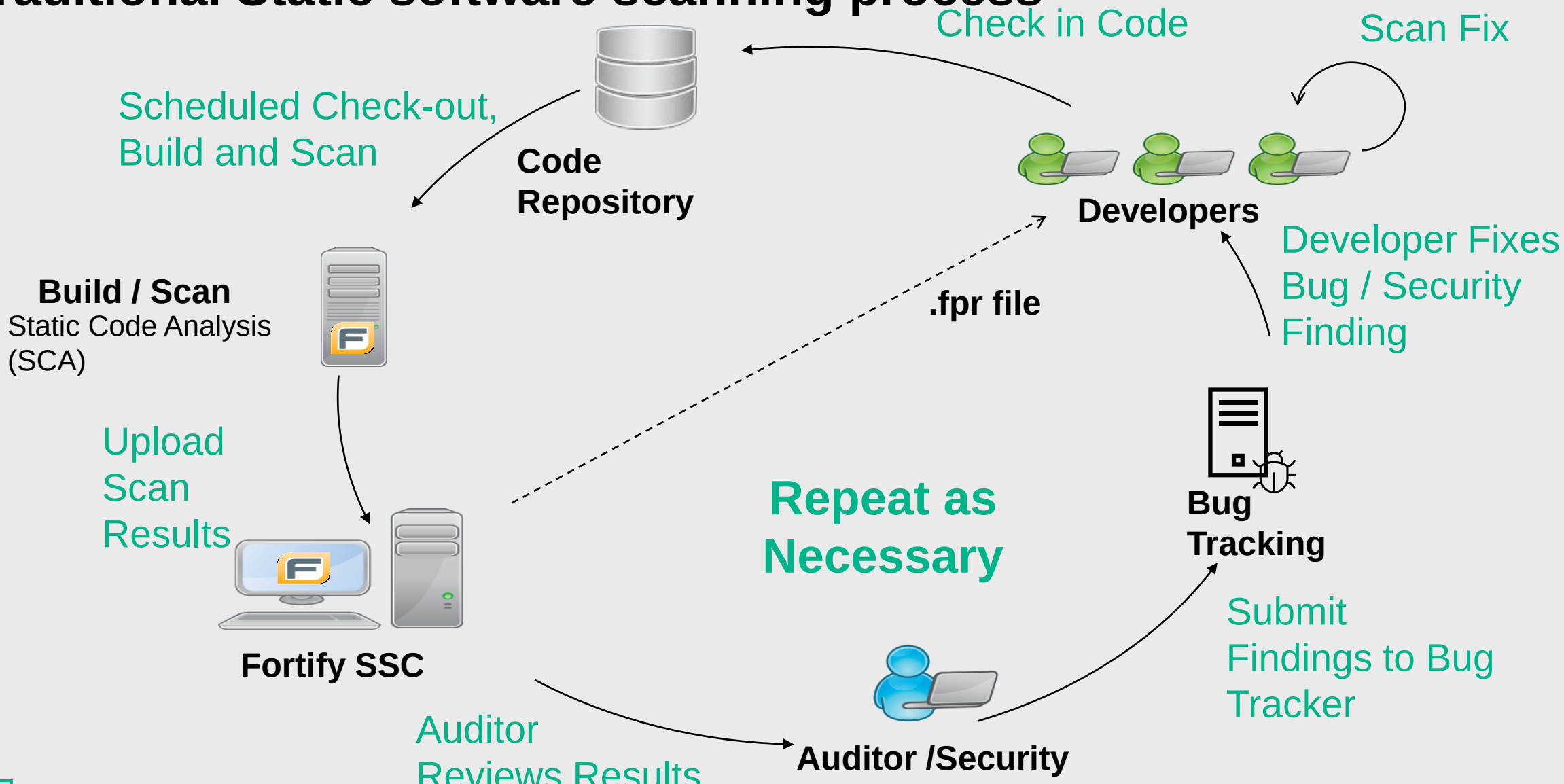
Developers need to move at the speed of business innovation



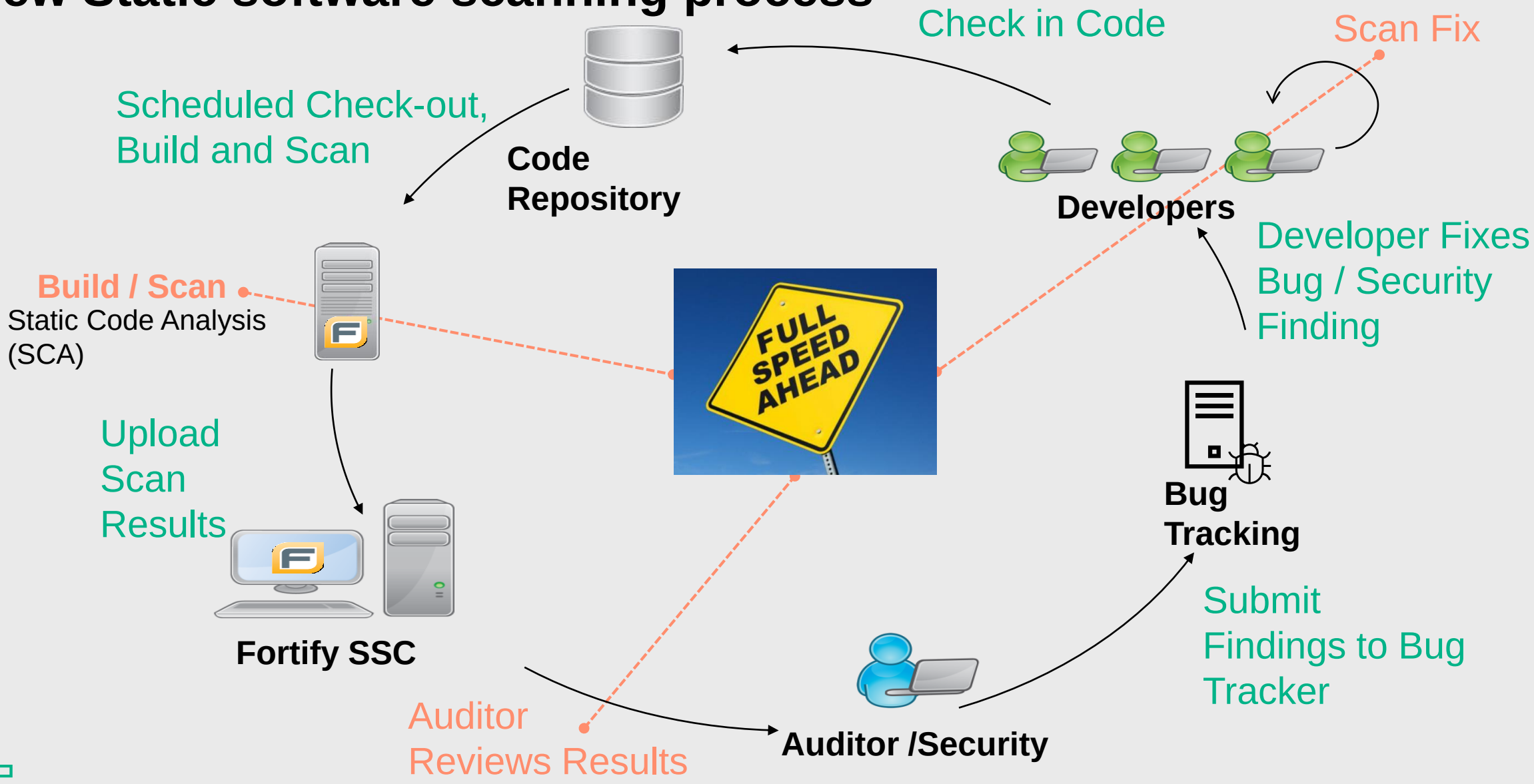
Steady Stream of Improvement

Thanks to consumerization, users now expect continuous improvements to apps rather than the traditional annual mega-updates

Traditional Static software scanning process



New Static software scanning process



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

Static Analysis Solution – Lengthy / Memory Intensive Scans

- Take advantage of multiple cores / processors / cloud
  - New in 17.10, use –mt flag for multi-threading
- Offload the scan from the build server to a dedicated scanning server
  - Mobile build or Cloudscan (Distributed scanning)
- Create scalable static scanning solutions
- Reduce the frequency of scans if necessary (determine how often meaningful changes to the code base are checked in)
  - Integrate into Build environment and embed appropriately
- Large enterprise applications should sometimes be broken up into logical modules based upon data flow
  - Bite-sized scanning
- Take advantage of lightweight static security scans early in the dev lifecycle
  - Incremental scanning

Multi-Threading scans

Why?

– Speed up scans

– Utilize full potential of the available resources with native java management

What?

– Significantly decrease the scan time.

– Support Agile/DevOps ecosystem.

How?

– Simply add the –mt flag to your command line scans

Incremental static analysis

Why?

– Customers are moving towards continuous integration and continuous builds making it difficult/impossible to always perform a full scan.

– Software is large and complex. Full scans take too long.

– Organizations are moving to Agile/DevOps tool chains and need scanning to fit in.

What?

– Significantly decrease the scan time.

– Support Agile/DevOps ecosystem.

How?

– Full scan to establish baseline.

– Subsequent scans can enable incremental mode.

– Incremental scans only assesses what has changed since last assessment.

– User can provide full source or subset of source. SCA will determine what has changed.

– SCA will measure the health of the incremental scans and recommend another full scan when needed.

Static Analysis Solution – Complex Builds

– The app sec team must work with / have access to developers / build engineers when automating / ensure it compiles before you get started

– Provide static integration examples on a internal wiki in addition to normal tool documentation to allow dev teams to be proactive

– Consider a centralized scanning solution outside of the build process if the integration and maintenance of build / security scripts becomes too much

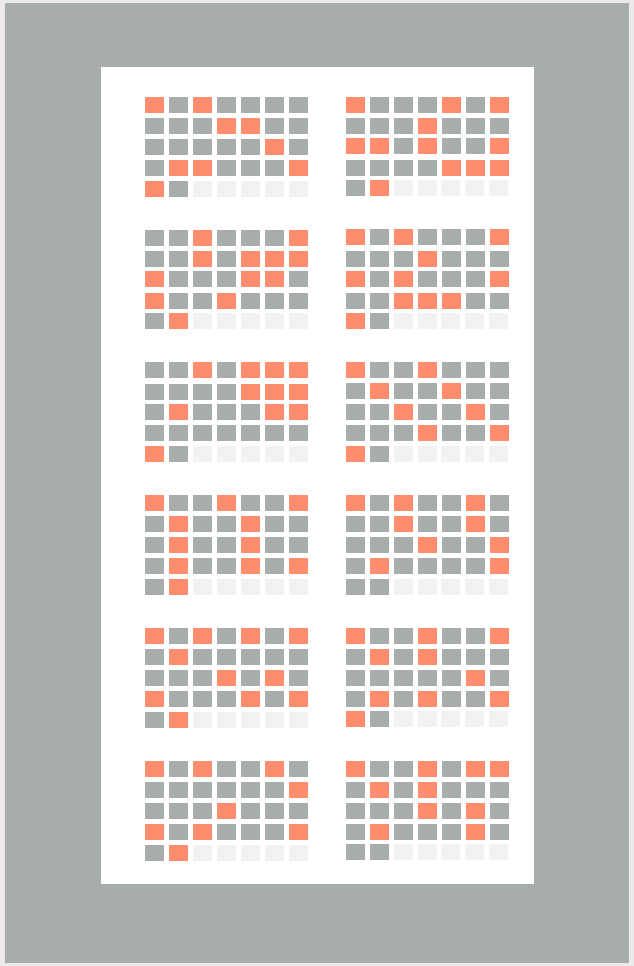
Hewlett Packard Enterprise

Static Analysis Solution – Build Frequency

– When building / scanning multiple times a day as a part of your CI process, ensure your storage solution is scalable

– Reasonable data retention policy should be applied to scanning result files
  – How valuable are these three year old scans?

– Automated merging of new scans with previous scans is a requirement to preserve previous audit decisions / trending



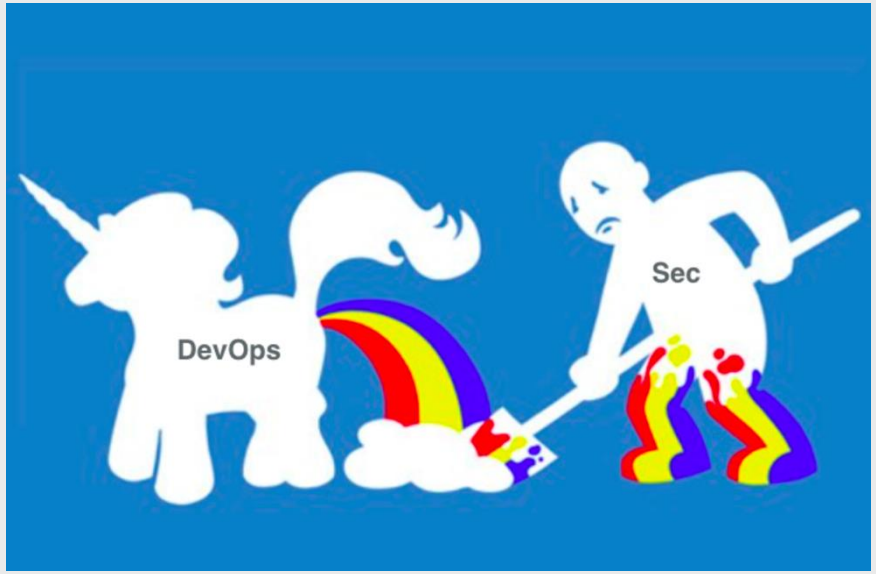
Hewlett Packard Enterprise

Static Analysis Solution – Triaging Static Findings

This is the main blocker of effective static security scans moving at a high speed

The quickest way to derail static testing is to push garbage findings to the dev team

- Security cannot be responsible for auditing static findings if they do not have the appropriate development background
  - Filling out exception paperwork filled with unimportant findings is a waste of everyone's time
- Development has to be accountable for acceptable organizational risk as they may be the auditor in many DevOps scenarios
- Vulnerability training needs to be available and current



Static Analysis Solution – Triaging Static Findings (continued)

– When auditing: Sort by common sources and sinks for dataflow issues

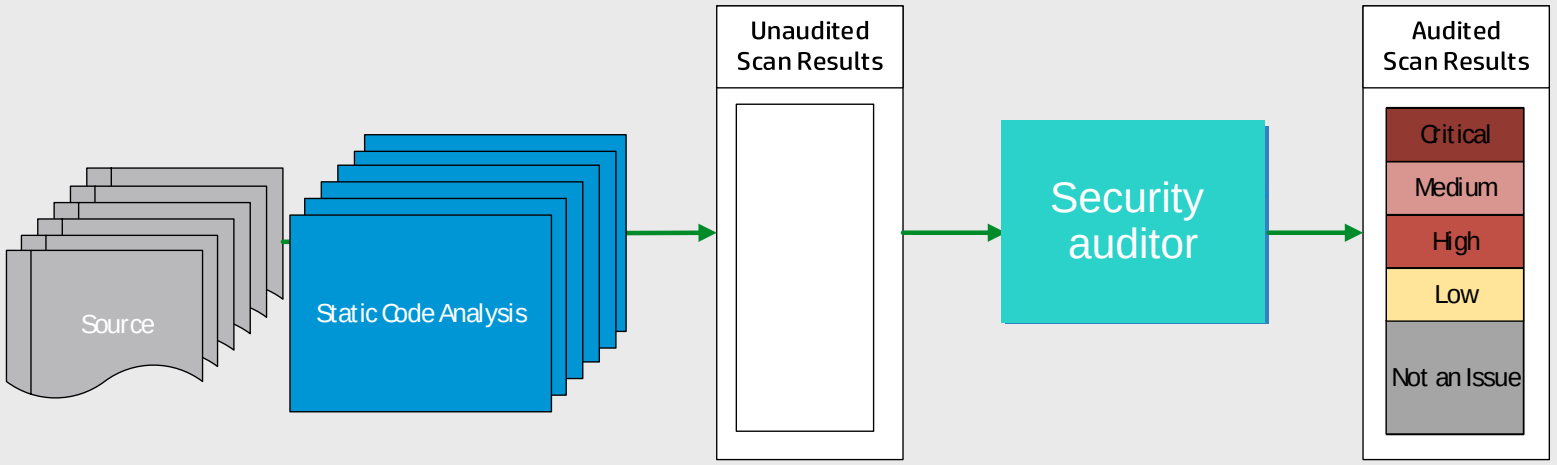
– Apply audit knowledge from past decisions

  – Start with a targeted list of vulnerabilities and expand that list as your program matures

– Make previously audited scan files available to anyone responsible for auditing

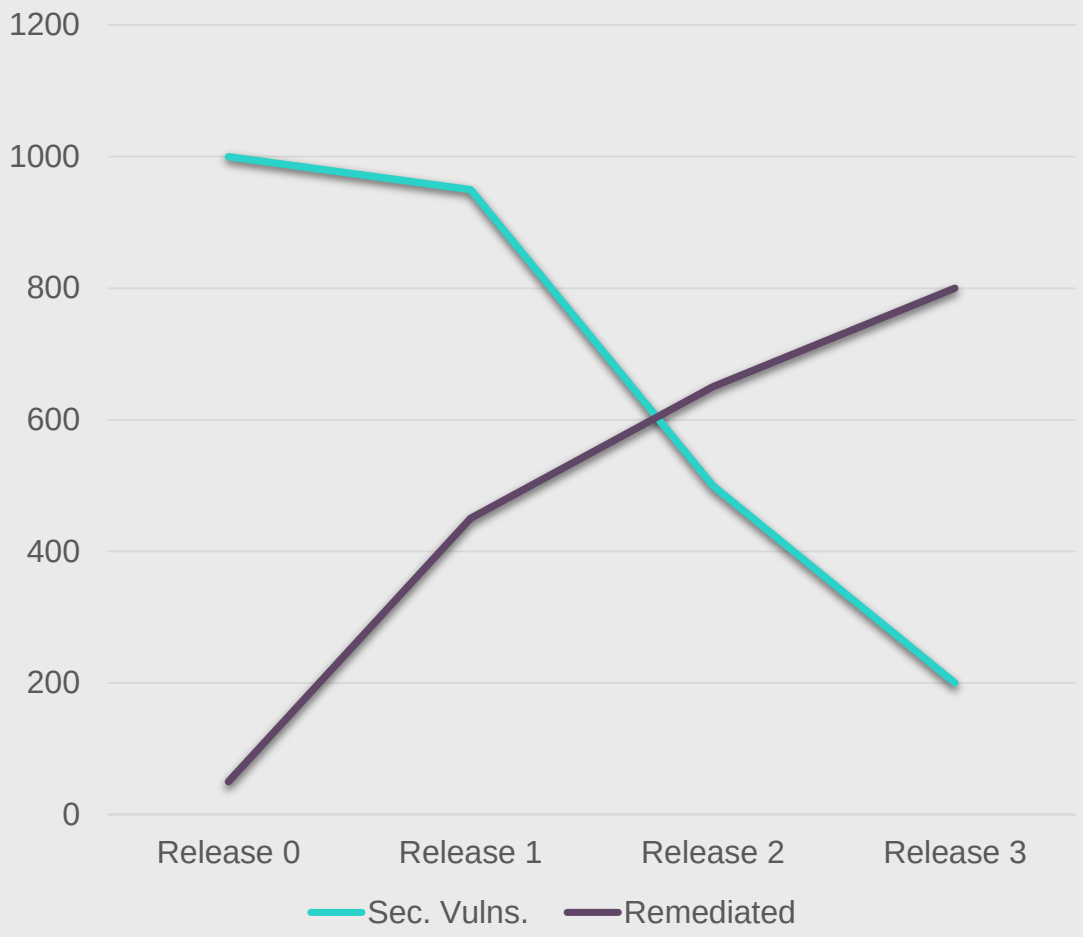
– Audit peer review

– Future: Machine learning to apply past audit decisions to predict future audit decisions



Static Analysis Solution – Remediation

- It’s time to fix something!
- You need to define effective security controls that work for your organizations specific technology stack
  - Talk to a software architect (if s/he is friendly)
  - Not every fix is created equally
  - Automate these recommendation via security controls that are language / technology specific
  - Internal security libraries for common languages can be very helpful



Static Analysis Solution – Communication

- Don’t make the developers go to a separate portal if they already have a bug tracking solution
- Do automate batch bug submission of security defects once findings are validated
  - Don’t submit bugs for unaudited findings
  - Don’t submit duplicate bugs
  - Don’t break builds for every unaudited static finding
- Understand your specific static analysis tools confidence thresholds and use that for automation
- Do mark builds as unstable / break if critical high confidence findings are flagged during a build
  - Requires a baseline scan of that application and audit to establish
- Understand what defect tracking solutions exist in your organization and understand the work involved to support them effectively, or decide not to and provide an alternative for security defects

Static Analysis Solution – Metrics

– I have never worked as on-site app sec SME where I didn’t have to provide tailored regular metrics to leadership

– When starting a security program, positive trending metrics of certain groups make adoption easier throughout the organization

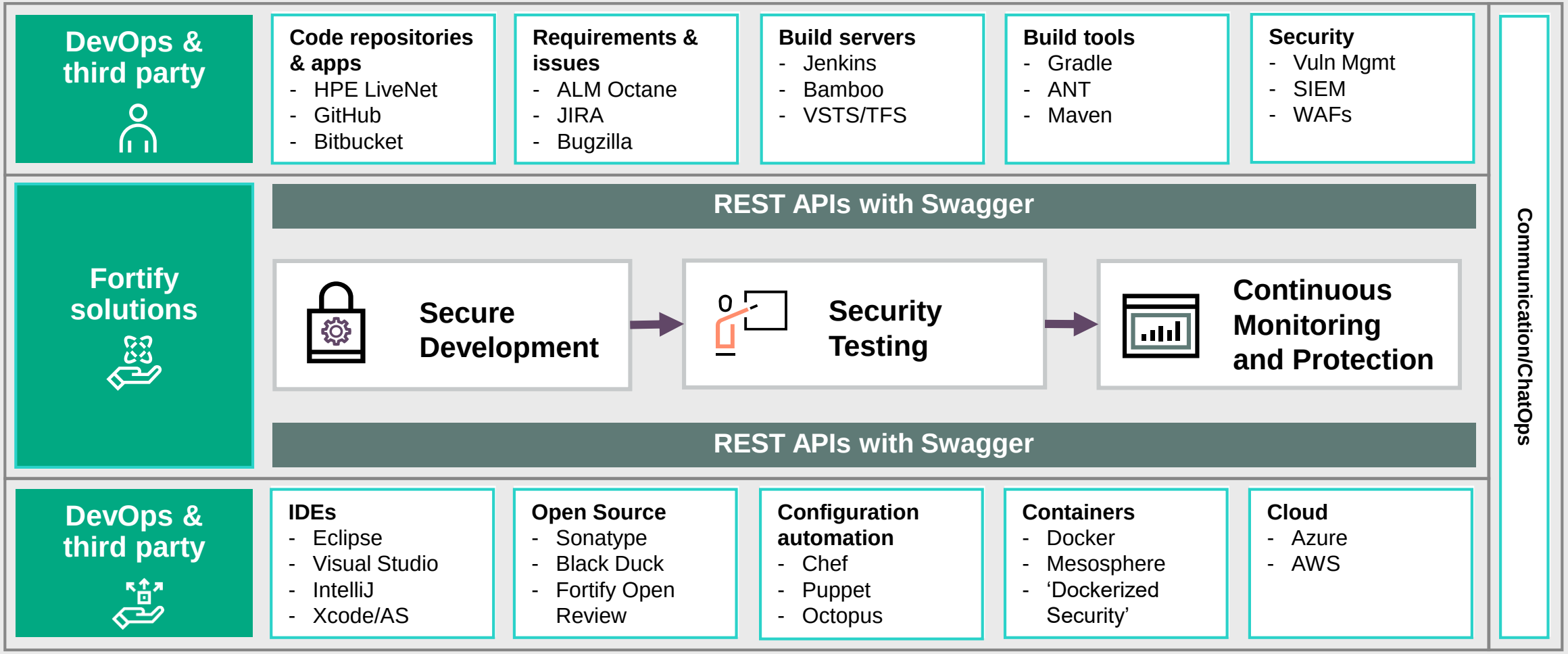
– Automate reporting and upload to src repo as they are a required artifact

– Tailor reports / dashboard to your organizations specific needs

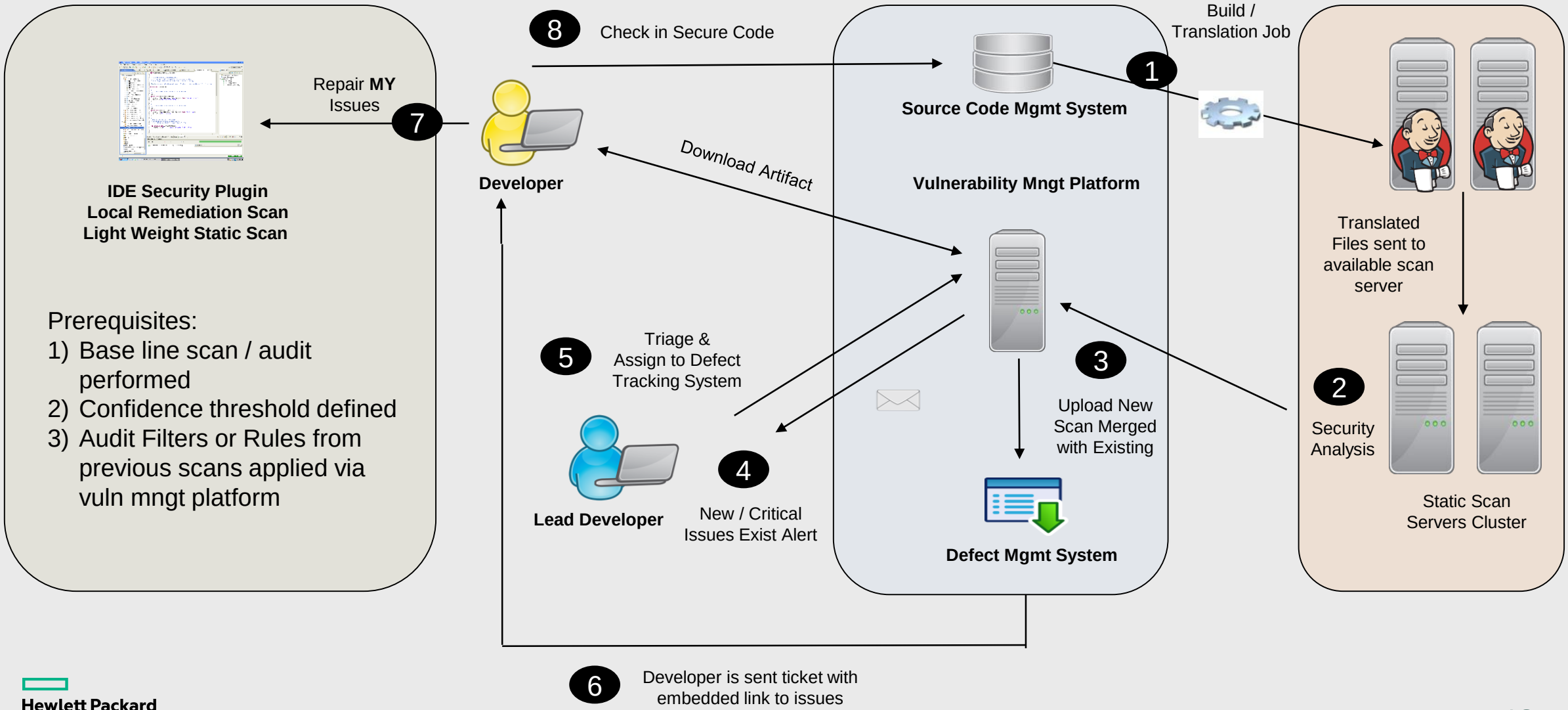
– Take advantage of other tools that may be available (GRC, etc)

Integrate

Securing DevOps through the Fortify Ecosystem integrations and automation



Effective / High Velocity DevSecOps - Example



Demo

This was a live demonstration of using Jenkins to perform a build and scan of code automatically. It was configured such that the build would be marked unstable if a Critical or High Fortify finding was found in the FPR file.

For more information please contact your Fortify Representative.

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 V4

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 deduplicated

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