Centers for Medicare & Medicaid Services
Information Security and Privacy Group

CMS Security Automation Framework (SAF) DevSecOps Best Practices Guide

Final Draft

Version 1.1

January 2021

Record of Changes

Version	Date	Author / Owner	Description of Change	CR #
0.1	10/19/2018	MITRE	Initial Draft	
1.0	5/16/2019	MITRE	Final Draft	
1.1	1/2021	MITRE	Updated Draft	

CR: Change Request

Executive Summary

CMS Programs leverage Agile and DevOps software development methodologies to support the continuous integration and continuous delivery required for their business solutions. At the same time, healthcare systems continue to be a primary target for bad actors due to the sensitive nature of CMS data. DevSecOps accelerates delivery by automating the required security and privacy processes for threat modeling, generating security and privacy documentation artifacts, change and source control management, static and dynamic code analysis, infrastructure hardening and least functionality checks.

This document describes proposed CMS best practices (e.g., standards, processes, and technologies) to ensure that CMS trusted applications and solutions are securely developed and continuously delivered to end users.

DevSecOps Best Practices include:

- **Security Validation as Code** – Testing standards, testing content (code), and automation tools to effectively know "is it secure?"
- **Documentation as Code** – Testing standards, testing content (code), and automation tools to effectively know "how am I secure?" to help maintain System Security Plan (SSP) documentation.
- **Change Management Auditing** – Processes to foresee significant security testing changes in a Sprint (Security Impact Analysis), and pipeline auditing to track unauthorized changes during builds. Answers the question: "what changed?"
- **Reporting** – Reporting and integration requirements to comply with stakeholder use of security data from the DevSecOps lifecycle. Stakeholders include developers, Information System Security Officers (ISSOs), Cyber Risk Advisors (CRAs), Adaptive Capabilities Testing (ACT) team, CMS Cybersecurity Integration Center (CCIC), and Federal Information Security Modernization Act (FISMA) reporting teams.
- **Operational Analytics** – Best practice process to engineer application audit log triggers during development to detect anomalies during operations and use this data to adapt to and plan for the next application development Sprint.
- **DevSecOps Process Improvement** – Describes what to measure and how to analyze the data to constantly improve the project's DevSecOps process. Improve future builds using metrics and measures of security debt, unauthorized changes during development, and detection of anomalies during operation.

Table of Contents

List of Figures

List of Tables

This page is intentionally blank.

1. Introduction

1.1 Background

CMS Programs have adopted Agile and DevOps software development methodologies to enable the continuous integration and continuous delivery of their business solutions. Healthcare systems remain a primary target for hackers because of the sensitive nature of CMS data. DevSecOps accelerates delivery by automating the required security and privacy processes for threat modeling, generating security and privacy documentation artifacts, change and source control management, static and dynamic code analysis, infrastructure hardening and least functionality checks.

1.2 Purpose

This document provides CMS DevSecOps best practices (e.g., standards, processes, tools, measures/metrics, and technologies) to ensure projects continuously deliver secure and trusted CMS applications and solutions.

DevSecOps Best Practices scope include:

- **Security Validation as Code** – Testing standards, testing content (code), and automation tools to effectively know "is it secure?"
- **Documentation as Code** – Testing standards, testing content (code), and automation tools to effectively know "how am I secure?" to help maintain System Security Plan (SSP) documentation.
- **Change Management Auditing** – Processes to foresee significant security testing changes in a Sprint (Security Impact Analysis), and pipeline auditing to track unauthorized changes during builds. Answers the question: "what changed?"
- **Reporting** – Reporting and integration requirements to comply with stakeholder use of security data from the DevSecOps lifecycle. Stakeholders include developers, Information System Security Officers (ISSOs), Cyber Risk Advisors (CRAs), Adaptive Capabilities Testing (ACT) team, CMS Cybersecurity Integration Center (CCIC), and Federal Information Security Modernization Act (FISMA) reporting teams.
- **Operational Analytics** – Best practice process to engineer application audit log triggers during development to detect anomalies during operations and use this data to adapt to and plan for the next application development Sprint.
- **DevSecOps Process Improvement** – Describes what to measure and how to analyze the data to constantly improve the project's DevSecOps process. Improve future builds using metrics and measures of security debt, unauthorized changes during development, and detection of anomalies during operation.

1.3 Scope

This guidance applies to all CMS FISMA applications processing CMS data for CMS missions, developed using a DevOps methodology. This guidance strives to build security into DevOps environments to mature to a DevSecOps methodology.

1.4 Audience

All system developers and maintainers (SDMs), CMS Business Owners, CMS ISSOs operating under an Agile/DevOps methodology; and supporting Information Security and Privacy Group (ISPG) Cyber Risk Advisors, Privacy Advisors, CCIC security monitoring team, ACT team, and ISPG front office. ISPG supports the CIO though recommendations for Authorizations to Operate (ATO).

1.5 Document Organization/Approach

This document is organized as follows:

Section	Purpose
Section 2: Goals and Objectives	Identifies high-level goals, objectives, alignments and strategies to ensure a comprehensive solution to enable Agile/DevOps environments to produce secure code and applications. Also identifies any constraints to be managed as they pose risk to the objectives needed to enable Agile/DevOps environments.
Section 3: Exemplar DevSecOps	Defines the best qualities of an exemplar DevSecOps methodology and a maturity model for reaching this exemplar.
Section 4: DevSecOps Best Practices	Discusses the current and future focus areas to help achieve the exemplar DevSecOps methodology.
Appendix A	NIST SP 800-53 Security Control Coverage Details
Appendix B	Security Control Mapping to SANS Top 25 CWE
Appendix C	Security Control Mapping to OWASP Top 10
Acronym List	Defines the acronyms used in this document

2. Goals and Objectives

The CMS DevSecOps vision is to enable the rapid release of trusted CMS business applications with

- Defined effective and approved DevOps processes
- Reduced overall time to receive initial ATO
- Continuous security data available Sprint-to-Sprint to maintain an ATO
- Automated security scanning, monitoring, testing and artifact generation
- Security and privacy risks and issues identified and resolved early in development

2.1 Benefits of DevSecOps

- **Built-In Security and Privacy.** DevSecOps encourages Business Owners to satisfy security and privacy requirements as part of their daily DevOps pipeline.
- **ATO Ready.** This integrated process rapidly delivers a more secure solution that satisfies many of the criteria needed to receive an ATO during development rather than at the end of development.
- **Automation.** Use of automation eliminates manual tasks, frees staff to work on unique problems, and can reduce the total pipeline time to deliver a trusted solution. Examples of areas for automation include security scanning, monitoring, and testing.
- **Issues Addressed Earlier.** Identifying and removing security and privacy risks and issues earlier saves resources and time.

2.2 Constraints

The following constraints must be managed as they pose risk to the objectives needed to enable Agile/DevOps methodologies:

- Workforce risks due to shortage of skilled DevSecOps staff
- Integration risks due to lack of coordination across all related development and operational areas, including security and privacy

3. Exemplar DevSecOps

DevSecOps requires a culture of trust and collaboration among development, operations, and security teams. Figure 3-1 illustrates how DevSecOps streamlines the delivery of secure solutions. The Defining Phase incorporates security and privacy requirements in the system architecture and design. During the Solving Phase, developers conduct Sprints that include security and privacy testing to identify vulnerabilities early in the development pipeline promoting proactive, incremental corrections prior to merging changes into the main build. Iteratively removing issues within each Sprint mitigates security and privacy technical debt to better position the system for initial ATO. The deployed system moves into continuous monitoring during the Delivering Phase, with fully automated processes to help manage the security and privacy posture.

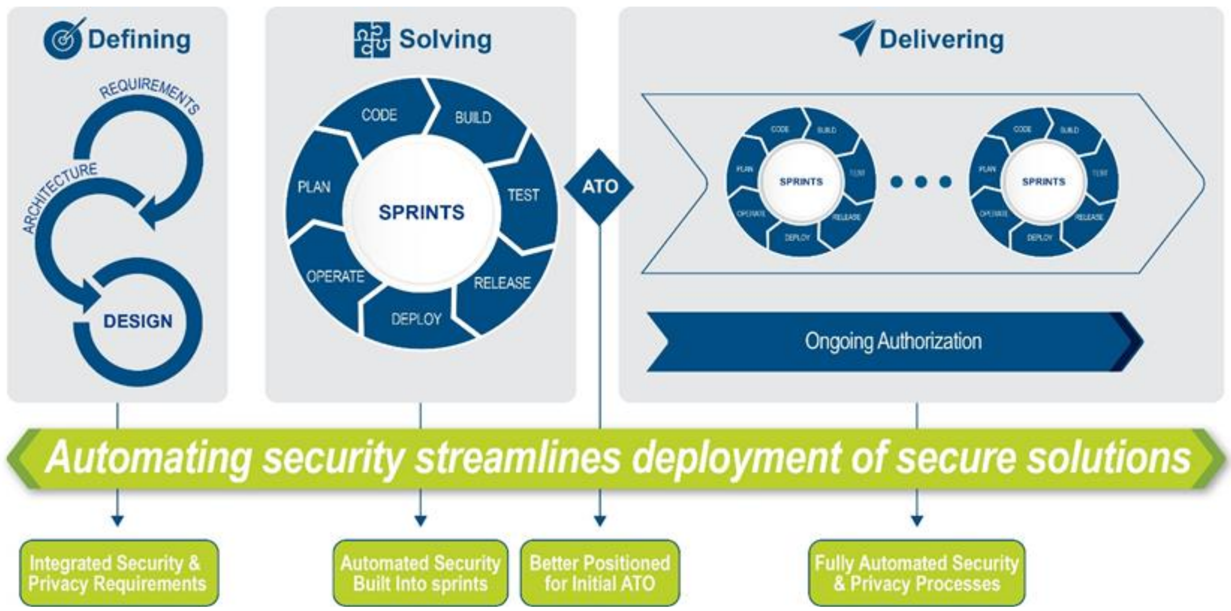

Figure 3-1 Exemplar DevSecOps Methodology

3.1 Top Qualities of DevOps

The primary quality that distinguishes a DevOps framework from all other methods of building and operating an application is: <u>repeatability</u>. All steps used to build and operate the application are recorded, so building the application again using the same steps will yield the same result. This supports the ability to troubleshoot root cause of detected defects by minimizing the number of variables introduced during a development cycle.

Building and operating an application under a DevOps framework:

1. Uses repeatable processes
   - all steps are recorded for re-use
2. Tests <u>everything</u>
   - shows if you've built it right, at each step

3. Automates everything
   - triggers the building and testing of as many steps as possible
4. Uses trusted materials (e.g., hardware, software, source code) and teams
   - the right materials and people to do the building and operating
5. Uses metrics to improve
   - measures to determine if the application operates properly during its production phase

These qualities are important to ensure consistency, reliability, and efficiency in how a business unit build and operates a CMS application.

3.2 Top Qualities of an Exemplar DevSecOps Framework

Security adds to the DevOps framework by:

1. Providing repeatable and re-usable processes to ensure security is built-in to the application
2. Providing repeatable tests to ensure the application is built securely, at each step
3. Automating as many security tests as possible
4. Trusting the materials and staff required to build and operate the application, from a security perspective
5. Providing metrics to know whether security is maintained each time the CMS application is built or operated

3.3 Value of Building Security into DevOps

- For the Business Owner:
  - CMS projects will have a **streamlined development pipeline** for delivering **trusted** services to end users
- For the Developer:
  - Empower the **developer to reduce the security defect debt**, Sprint-to-Sprint
- For the ISSO:
  - Enable the **ISSO to always know the application's security posture**
- For ISPG:
  - Use **DevOps pipeline to automatically generate security testing data**
    - Security controls and testing are embedded within the development pipeline, resulting in faster delivery of secure solutions
    - Consistency and transparency in the way tests are performed
    - **ISPG** will provide a **library of the specific tests to be used**
  - Consistency and consolidation of the results from those tests

    - ♦ **ISPG** will **approve the format of results**, to promote consistency across all environments.
  - Lay a foundation **for moving to ongoing authorization** decisions.

3.4 Benefits of Building Security into DevOps

- For the Developer:
  - Higher security assurance, always, of the **security of their code as well as the supporting stack**
  - **Lower security debt** (i.e., higher security defect remediation rates)
- For the ISSO:
  - **Always aware of security status**
  - Faster security go-live decisions during Sprints
  - Timely, clear, concise, consolidated, trusted security data
  - Clear understanding of which security controls are either directly supported or validated as in place
- For the Business Owner:
  - **Reduced assessment time and cost by using DevOps security data**
  - **Moves CMS toward ongoing authorization**
  - Use change management auditing data to perform business analytics for non-security use cases (e.g., Sprint planning efficiency)
- For ISPG:
  - **Confidence in the security tests and results** (by maintaining testing code and testing standards)

3.5 Exemplar DevSecOps Components

3.5.1 Standards

- Secure coding to avoid defects based on the following standards:
  - Common Weakness Enumeration (CWE)/SANS Top 25 Most Dangerous Software Errors[1]
  - Open Web Application Security Project (OWASP) Top 10: The Ten Most Critical Web Application Security Risks[2]
- Configuration security settings hardening (in order of CMS preference, as available)
  - DISA Security Technical Implementation Guides (STIGS)[3]

[1] https://cwe.mitre.org/top25/
[2] https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project
[3] https://public.cyber.mil/stigs/

  - Center for Internet Security (CIS) Benchmarks[4]
  - Vendor recommendations
- Least functionality
  - Standards to minimize the attack surface of an application or its infrastructure by eliminating unnecessary functions, services, ports, protocols, etc.
- Patching
  - Ensuring that all infrastructure is up to date on all security-related patches
- Recommended Metrics
  - All high security defects and 90% of all medium or low security defects are resolved before allowing affected functionality to be deployed to production
  - No security defect may carry-over unresolved through more than two Sprints
  - Unplanned (unauthorized) changes for any Sprint is less than 5% of planned (authorized) changes.
  - Time to receive initial ATO for new systems is 25% less than equivalent new systems
- Triggers for significant changes
  - Supports decisions for modified or additional testing based on each Sprint's initial security impact analysis

3.5.2 Processes

- Development
  - Training developers in secure coding
  - Secure code review processes
  - Configuration management, including review/approval process for standards
    - Security hardening
    - Patching
    - Least functionality
  - Sprint-level security oversight for Sprint-to-Sprint go-live decisions

3.5.3 Technology

- Change planning/tracking (e.g., Jira)
- Source code repository (e.g., GitHub)
- Orchestration (e.g., Jenkins, Ansible, Terraform)
- Automated testing (e.g., InSpec, operational scans)
- Log aggregation (e.g., Splunk)
- Analytics
  - tracking unauthorized changes during development (change management auditing)

[4] https://www.cisecurity.org/cis-benchmarks/

– application audit log trigger during operation to detect anomalies to adapt to and plan for the next application build

3.5.4 Authorized Pipeline/Process Fidelity

To meet the goals, objectives, and qualities of the above exemplar DevSecOps framework, the DevSecOps pipeline shall be required to meet an acceptable level of reliability and trustworthiness. The crucial goal for the pipeline is to produce a secure CMS application while generating the necessary security data required to maintain authorization initially, and at the end of each Sprint. Table 1 illustrates a maturity checklist for business owners. For security in DevOps to work for production changes, CMS will approve DevSecOps pipelines and processes complying with this checklist:

- Note: This is a checklist to include Security into DevOps pipelines/processes. The prerequisite for compliance with this checklist is a DevOps environment with a fully automated CI/CD pipeline, and no manual user interaction beyond committing software into the repository.
- Primary Security Goal(s): Maintain security debt at a consistent level, Sprint to Sprint, by enabling developers and ISSOs to verify security and compliance early and often during each Sprint. Automation and standardization of this security data is essential.

Table 1 - DevSecOps Pipeline/Process Checklist

1. Does the pipeline automatically validate, at each create and configure for each build, **security configuration settings compliance** of underlying application stack components?
   Evaluates supporting cloud, network, operating system, database, app-server and web-server components' configurations against STIGs, CIS Benchmarks and CCE compliance.
2. Does the pipeline automatically validate, at each create and configure for each build, **security vulnerability levels** of underlying application stack components?
   Assesses software patch levels and CVE compliance.
3. Does the pipeline automatically validate, at each create and configure for each build, **least functionality** of underlying application stack components?
   Limits services, ports, and protocols for application stack to function, compliant with National Institutes of Standards and Technology Special Publication (NIST SP) 800-53 Security Control for Configuration Management: CM-7 Least Functionality requirements.
4. Does the pipeline automatically perform **static code analysis**, at each commit, against CMS application source code?
   Analyzes at least 95% of the lines of code (95% code coverage) and perform linting checks for security issues against, at a minimum, SANS Top 25 CWE compliance.
5. Does the pipeline automatically perform **dynamic code analysis**, at each create and configure for each build, against CMS application compiled/running code?
   Assesses code security against, at a minimum, OWASP Top 10 CWEs.
6. Does the pipeline automatically generate all of the above security data in the ISPG standard "Heimdall Data Format" for machine-readability, assigning severity levels to each security test result (critical, high, medium, low) and mapping all security test results to NIST SP 800-53 security controls?
   Ensure compliance with InSpec JSON output reporter schema including, at a minimum, these labels: title, description, check text, fix text, relevant NIST SP 800-53 tags and impact level for each defect.
7. Does the pipeline automatically track and compare planned versus executed changes, to prove that planned changes, and ***only*** the planned changes, were implemented during a Sprint?
8. Do developers and ISSOs certify that all high security defects and 90% of all medium or low security defects are resolved before allowing affected functionality to be deployed to production?
9. Do developers and ISSOs assure that no security defect may carry-over unresolved through more than two Sprints?
10. Are unplanned (unauthorized) changes for any Sprint less than 5% of planned (authorized) changes?

3.5.4.1 DevSecOps Checklist Crosswalk: Sprint Days 1-2

The ISSO works with Developers and DevOps team to:

- Identify all proposed functional changes (e.g., in project issue tracker) ***[Check #7]***

- Identify additional security testing required, for example, new/changed:
  - Code to add to scope of static code analysis tests ***[Check #4]***
  - APIs, web forms, web pages to add to scope of dynamic analysis tests ***[Check #5]***
  - Cloud, network, operating system, database, app-server and web-server components to add to scope of security configuration setting, patchable vulnerability, and least functionality tests ***[Checks #1, #2, #3]***
- Document the above in a draft Sprint Security Impact Analysis (SIA) ***[Check #7]***

3.5.4.2 DevSecOps Checklist Crosswalk: Sprint Days 3-8

Developers use the DevOps pipeline to help:

- At each commit: automatically perform **static code analysis** against application source code ***[Check #4]***
- At each create and configure for each build, automatically validate:
  - **security configuration settings compliance,** ***[Check #1]***
  - **security vulnerability levels,** and ***[Check #2]***
  - **least functionality** of underlying application stack components ***[Check #3]***
- At each create and configure for each build: automatically perform **dynamic code analysis**, against application compiled/running code ***[Check #5]***
- Resolve all high security defects and 90% of all medium or low security defects detected by the pipeline during the Sprint ***[Check #8]***

3.5.4.3 DevSecOps Checklist Crosswalk: Sprint Days 9-10

- The ISSO works with the Developers to review logs from the DevOps pipeline:
  - To compare executed changes to the planned changes documented in the Draft SIA and verify that planned changes, and *only* the planned changes, were implemented during a Sprint ***[Check #7]***
  - If the unplanned changes were not covered by security testing, the ISSO recommends the affected functionality be held back from production deployment until security testing can be performed ***[Check #7]***
  - The ISSO and Developers work to ensure that unplanned (unauthorized) changes for any Sprint is less than 5% of planned (authorized) changes ***[Check #10]***
- The ISSO reviews final-run security testing results to verify that all high security defects and 90% of all medium or low security defects are resolved before allowing affected functionality to be deployed to production. ***[Check #8]***
- Developers and ISSOs assure that no security defect may carry-over unresolved through more than 2 Sprints. ***[Check #9]***

4. DevSecOps Best Practices

The following sections discuss each best practice area in support of achieving and maintaining a DevSecOps pipeline, process, and technologies.

4.1 Current DevSecOps Best Practice Focus Areas

An underlying goal of DevSecOps (i.e., building security into DevOps) is preventing vulnerable applications from reaching production, Sprint to Sprint. However, faster deployments that are also part of DevOps, while appealing, can also lead to the deployments of vulnerable applications, leading to higher risk of unauthorized access to CMS data. For the ISSO, it is a tremendous challenge to track changes and weigh security at the end of each Sprint. For the developer, it is also a challenge to receive timely, concise security defect information each time they commit and build during a Sprint. The ISSO and developer need to be able to make an informed decision at the end of each Sprint to recommend a "security go-live," having the confidence to know that the application about to be deployed is secure. To do this, they need timely security data, of various types and prepared in specific ways.

The data needed by the ISSO and developer can be grouped into 3 basic types:

1. **"Is it secure?"** – data at all levels of the application stack proving that no high impact security defects remain, about 90% of all medium or low security defects have been resolved, and each security defect is mapped to the relevant NIST SP 800-53 security control (for context), across the following areas:
    - Secure Configuration Settings (i.e., STIG or CIS Benchmarks against all supporting cloud, network, operating system, database, app-server, web-server components, checking for Common Configuration Enumeration defects)
    - Vulnerability Scanning (i.e., Patch Levels, checking for Common Vulnerabilities and Exposures [**CVE**®] defects)
    - Least Functionality (limit services, ports, and protocols only for application to function, NIST SP 800-53 CM-7 Least Functionality defects)
    - Static Code Analysis (i.e., against CMS application source code, checking for CWE defects)
    - Dynamic Code Analysis (i.e., against CMS application compiled/running code, checking for CWE defects)
2. **"How is it secure?"** – data to supplement the System Security Plan to reflect the latest technical security system design
3. **"What has changed during this Sprint?"** – data that proves that the planned changes, and ***only*** the planned changes, were implemented during the Sprint

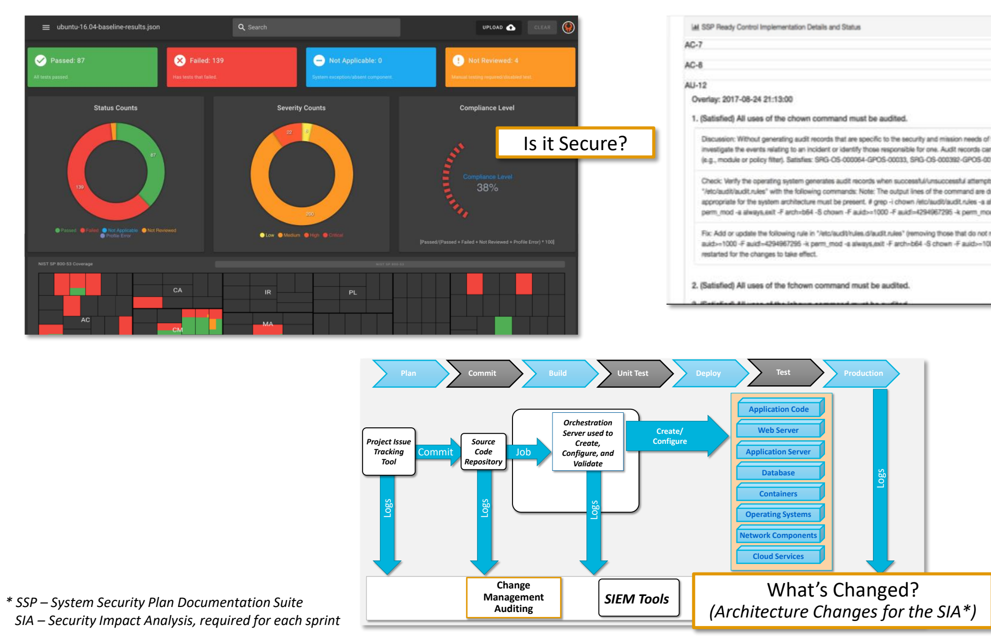

Figure 4-1 Security Data Needed by the ISSO and Developer each Sprint

The current DevSecOps best practice focus areas support the following user stories:

User Story / Pain Point: ***As a DevOps/Agile developer:***	**Best Practice Area**
*… it is **cheaper and easier to fix security defects early** in the development process. I need tools that are integrated into my CI/CD pipeline to help me assess and correct security defects on-demand, as I build, iteratively: to build, get **immediate security defect feedback**, determine root-cause, correct, and re-build many times a day…*	Security Validation as Code *(Is it secure?)*
*… and **automatically document the as-built security** in my system for my ISSO…*	Security Documentation as Code *(How is it secure?)*
*… and I need to **account for all of my changes** through the pipeline **before requesting a security go-live decision** for my system…*	Change Management Auditing *(What changed during this sprint?)*

Figure 4-2 User Stories Supported by Current DevSecOps Best Practice Areas

How can all this data possibly be provided every two weeks? Fortunately, DevOps pipelines embrace automation. Likewise, to provide timely data to developers and ISSOs, security must also embrace automation within the pipeline. The following figure illustrates the tests to be called by the pipeline to generate this security data automatically:

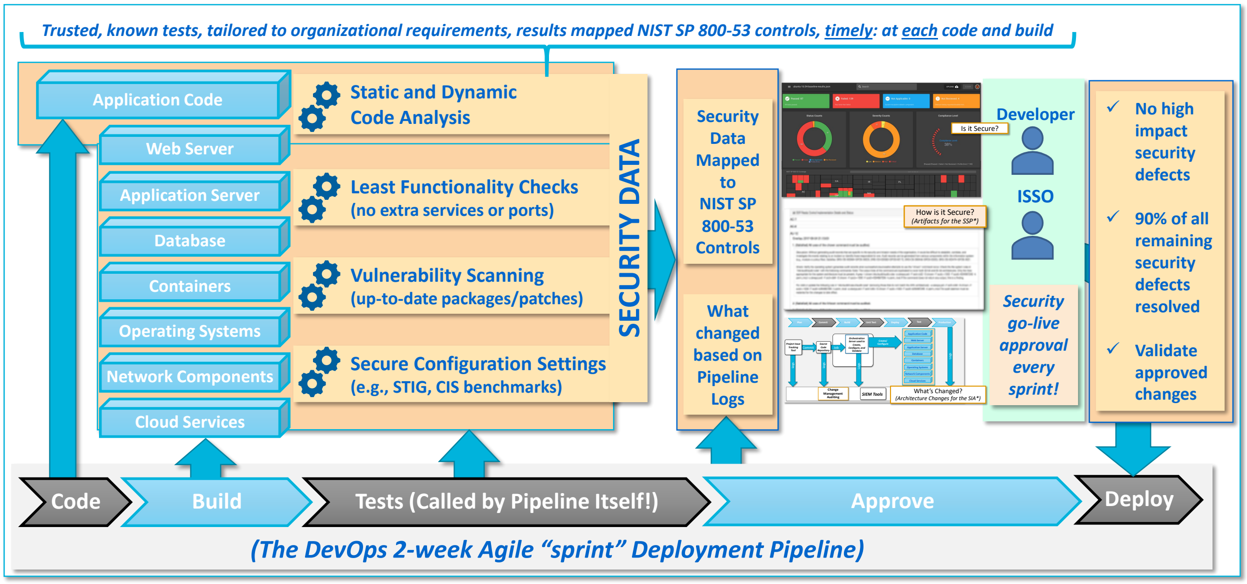

Figure 4-3 Automated Security Data needed from the DevOps Pipeline

CMS has embraced the Gartner "infinity loop" DevOps cycle[5] as a model for how teams should conduct development and operation of CMS applications. The figure below illustrates how the current best practice focus areas support this model:

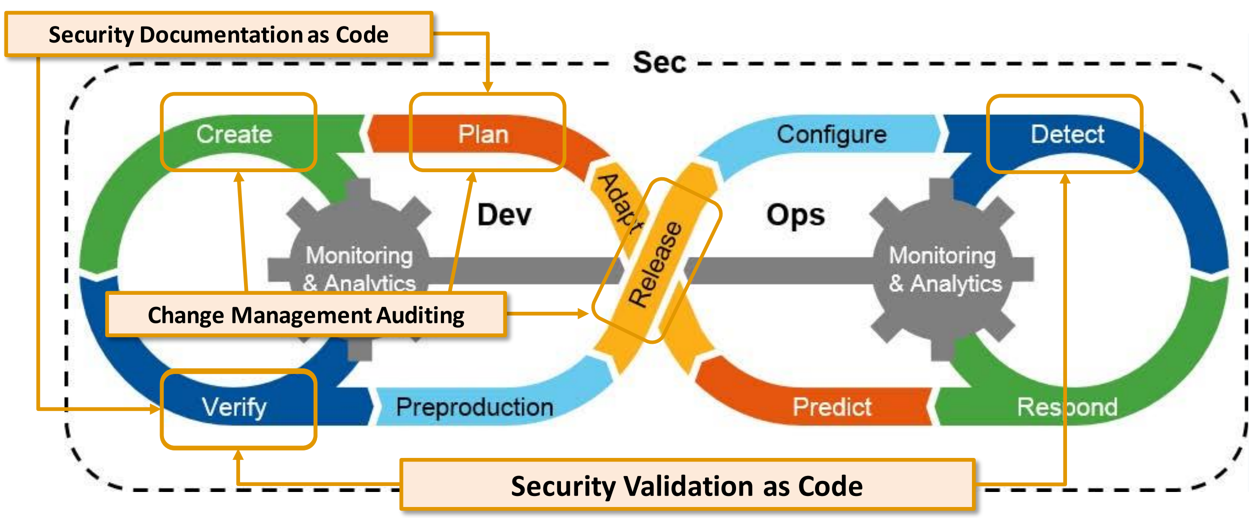

Figure 4-4 Current Best Practice Area Support for DevSecOps "Infinity Loop"

4.1.1 Security Validation as Code Best Practice

The Security Validation as Code focus area supports the user story for "Is it secure?":

[5] *Original "Infinity Loop" Graphic Source: Gartner - DevSecOps*

"As a DevOps/Agile developer, it is cheaper and easier to fix security defects early in the development process. I need tools that are integrated into my CI/CD pipeline to help me assess and correct security defects on-demand, as I build, iteratively: to build, get immediate security defect feedback, determine root-cause, correct, and re-build many times a day."

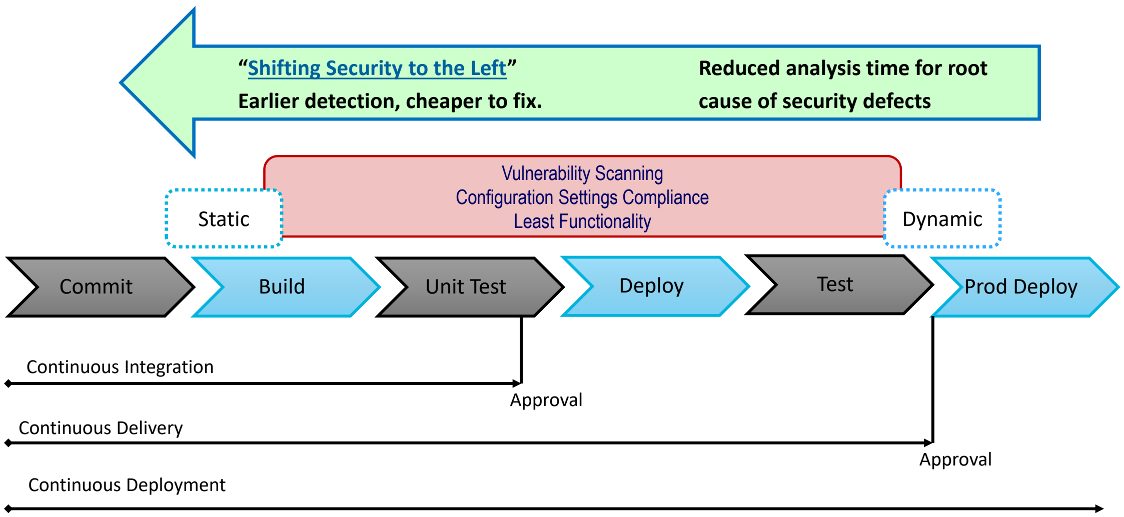

Figure 4-5 Shifting Security Testing "Left" to Fix Security Defect as Early as Possible

This focus area uses the pipeline to directly perform security validation tests. The security defects produced by these tests need to either already map their defect results to NIST SP 800-53 security controls or provide a category for the security defect that can easily be mapped to NIST SP 800-53 security controls automatically.

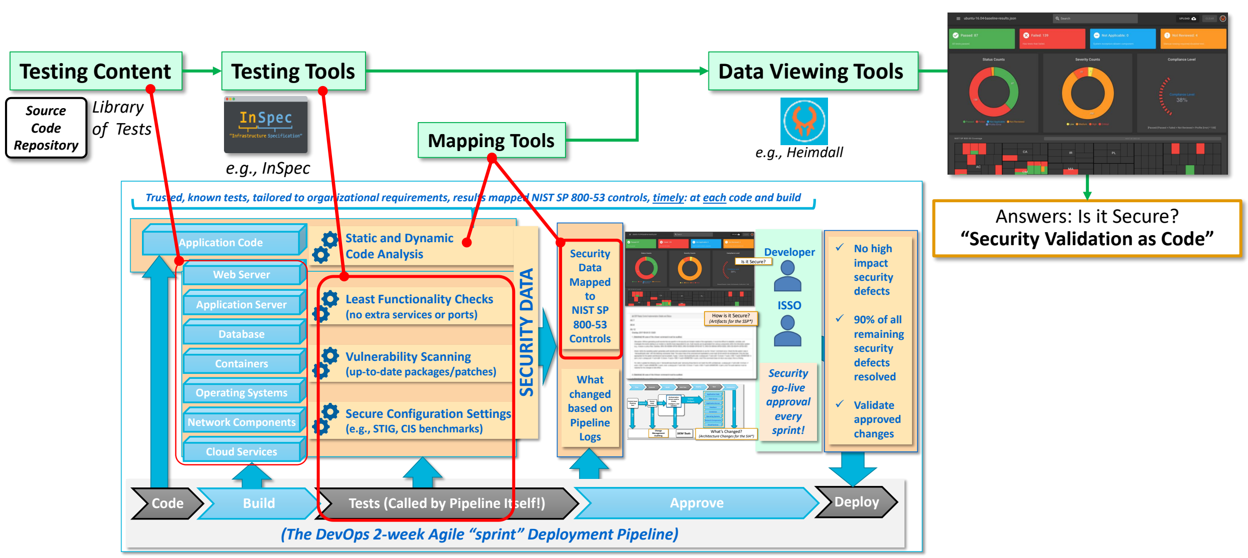

Figure 4-6 Technology Approach for Security Validation as Code Best Practice Area

4.1.1.1 Testing Tools

To perform least functionality, patching, and configuration setting checks against the underlying infrastructure of cloud, network, operating system, container, database, application, and web server technologies supporting an application, CMS has adopted InSpec[6], an open-source testing framework that can be incorporated into the existing testing harness of a pipeline's orchestration server. InSpec is well-suited to performing concrete sets of tests against known infrastructure components supporting an application.[7]

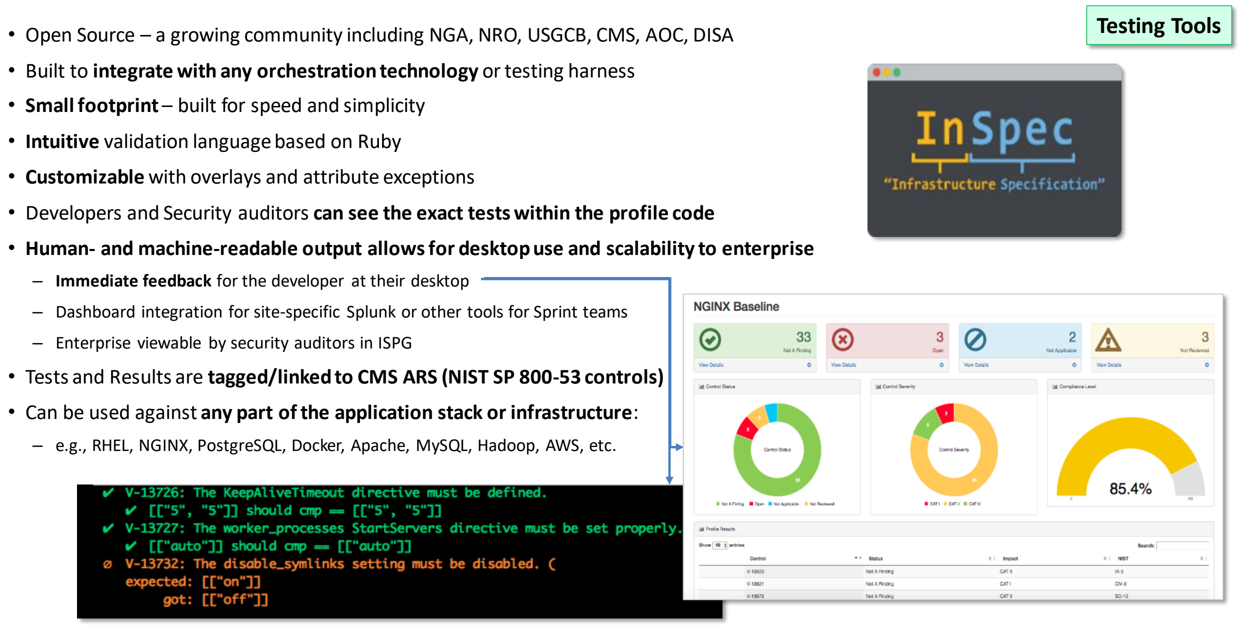

Figure 4-7 InSpec Open Source Testing Framework for Security Validation as Code Best Practice Area

[6] https://www.inspec.io/

[7] At this time InSpec isn't very well suited for automating static and dynamic analysis tests. Static and dynamic analysis tools are more complex in their modeling and analysis of custom application code. We discuss how a DevOps pipeline can handle output from common static and dynamic tools in section 4.1.1.3.

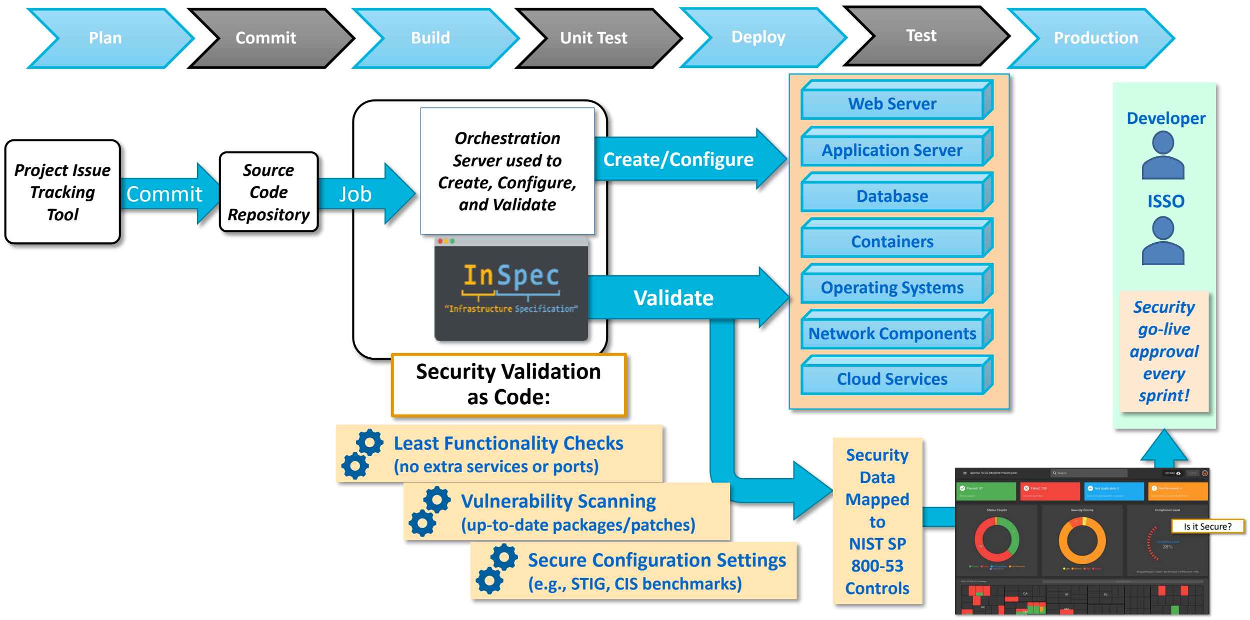

Figure 4-8 InSpec-based Secure Configuration, Vulnerability, and Least Functionality Validation Checks

4.1.1.2 Testing Content (Code)

InSpec relies on testing content in the form of InSpec "profiles" to perform validation. A library of approved profiles is expected to be available for all DevOps teams to use within their pipelines to validate the security of their underlying infrastructure, as often as needed, to reduce and maintain security debt. The current [CMS SAF library](https://saf.cms.gov/#/validate) is shown below, and is expected to grow as CMS adopts different technologies to support CMS applications:

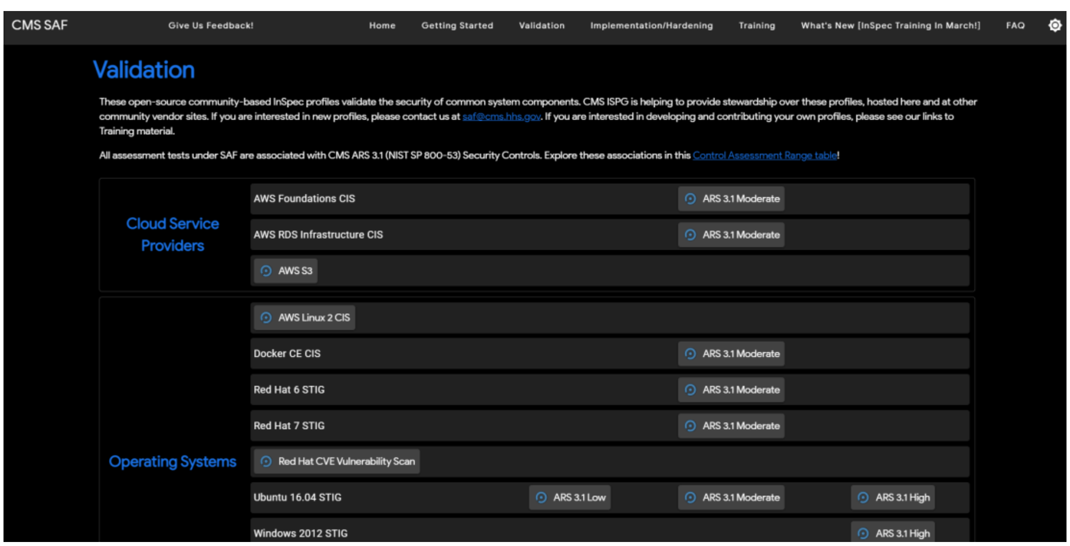

Figure 4-9 Testing Content for Security Validation as Code: Current InSpec Profiles[8]

[8] https://saf.cms.gov/#/validate

NIST SP 800-53 tagging: When CMS develops InSpec profiles, we ensure that each test has an associated NIST SP 800-53 tag.

```
control "V-71921" do
  title "The shadow file must be configured to store only encrypted
representations of passwords."
  desc  "Passwords need to be protected at all times, and encryption is the
standard method for protecting passwords. If passwords are not encrypted, they
can be plainly read (i.e., clear text) and easily compromised. Passwords
encrypted with a weak algorithm are no more protected than if they are kept in
plain text."
  impact 0.5
  tag "gtitle": "SRG-OS-000073-GPOS-00041"
  tag "gid": "V-71921"
  tag "rid": "SV-86545r1_rule"
  tag "stig_id": "RHEL-07-010210"
  tag "cci": ["CCI-000196"]
  tag "documentable": false
  tag "nist": ["IA-5 (1) (c)", "Rev_4"]
  tag "check": "Verify the system's shadow file is configured to store only
encrypted representations of passwords. The strength of encryption that must be
used to hash passwords for all accounts is SHA512.

Check that the system is configured to create SHA512 hashed passwords with the
following command:

# grep -i encrypt /etc/login.defs
ENCRYPT_METHOD SHA512
…
```

IA-5(1),
NIST SP 800-53's Security Control for
Identification and Authentication - Password-based Authentication

Figure 4-10 NIST SP 800-53 Security Control Context for InSpec Tests

When an InSpec profile is run against a target system, the security data output includes this tag as well as all other tags, description information and the code used for each test.

4.1.1.3 Data Mapping Tools to Adapt Data from Static and Dynamic Code Analysis Tools

At CMS, DevOps teams are adopting various static and dynamic code analysis tools. To help DevOps teams manage data from these tools, CMS is developing mapping algorithms and tools to ensure that the security defects found by commonly used static and dynamic code analysis tools are also mapped to NIST SP 800-53 controls, and viewable in the same fashion as InSpec output. Static code analysis tools are typically expected to be called by the source code repository (e.g., CMS GitHub) upon each major commit, and dynamic code analysis tools called by the existing testing harness of a pipeline's orchestration server.

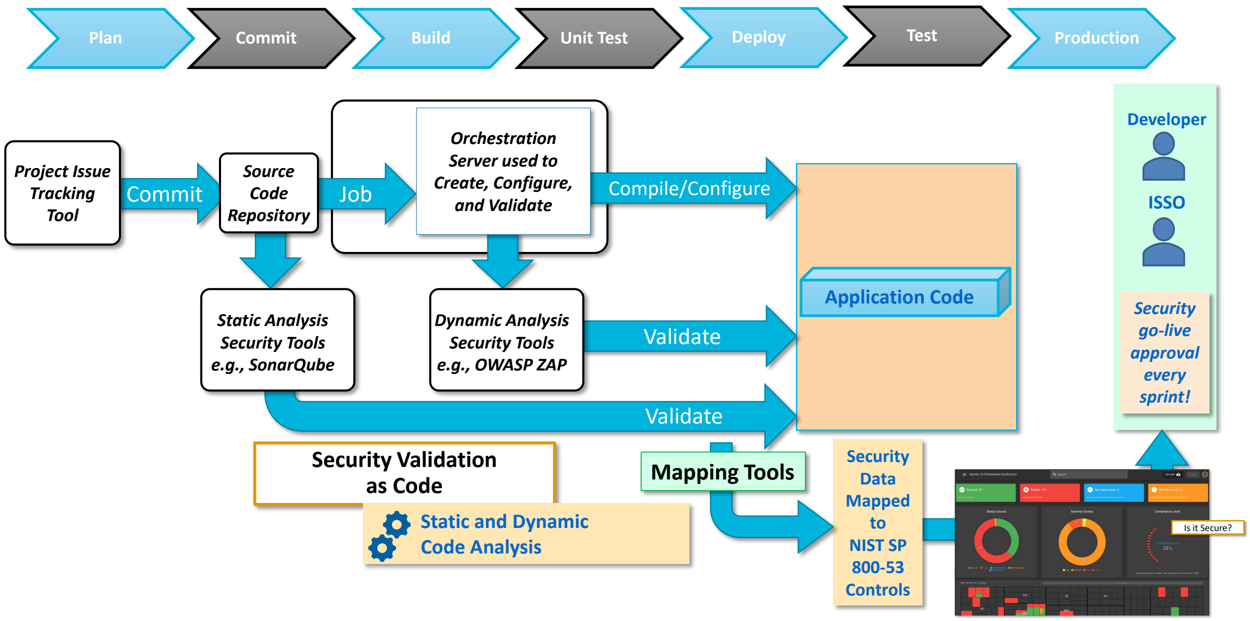

Figure 4-11 Data Mapping Tools for Static & Dynamic Code Analysis Tool Output

4.1.1.4 Data Standardization and Viewing Tools

In addition to ensuring the pipeline generates timely security data, CMS also requires the data to be prepared in specific, consistent ways, to meet several goals:

- **Machine-readability** – to ensure that the data can be moved and transformed as needed
- **Security control context** – security defects tagged to NIST SP 800-53 controls, to provide a common point of reference for the security context of any defect
- **Transparency** – shows not only the test code, but the reasoning behind it, all details of the security defects from the original testing profile or tool, and the explanation of how to fix the defect
- **Flexibility** – allows viewing at the developer's desktop from the command-line, or using a graphical user interface (GUI) to view on the developer's laptop, DevOps team server, or data center server
- **Consistency** – allows for a familiar view for a wide audience: developers, ISSOs, and Business Owners

Machine-readability is important to allow security data to be transported to, easily read by, and presented on a customer's preferred dashboard.

CMS uses an open-source tool called "Heimdall[9]" to tailor CMS viewing requirements for the machine-readable data produced by InSpec and data mapping tools, and uses CMS's existing Splunk infrastructure as a general-purpose transport to move this data for team and enterprise (e.g., data center) viewing. This helps the pipeline automatically generate all security data in a standard "Heimdall Data Format" for machine-readability, assigning severity levels to each

[9] https://heimdall-lite.cms.gov/, https://github.com/mitre/heimdall2, https://hdf-json-to-splunk.mitre.org/

security test result (high, medium, low) and mapping all security test results to NIST SP 800-53 security controls. The output is standardized in a json format including a title, description, check text, fix text, relevant NIST SP 800-53 tag and impact level for each security defect.

The output is standardized in a json format including a title, description, check text, fix text, relevant NIST SP 800-53 tag and impact level for each security defect.

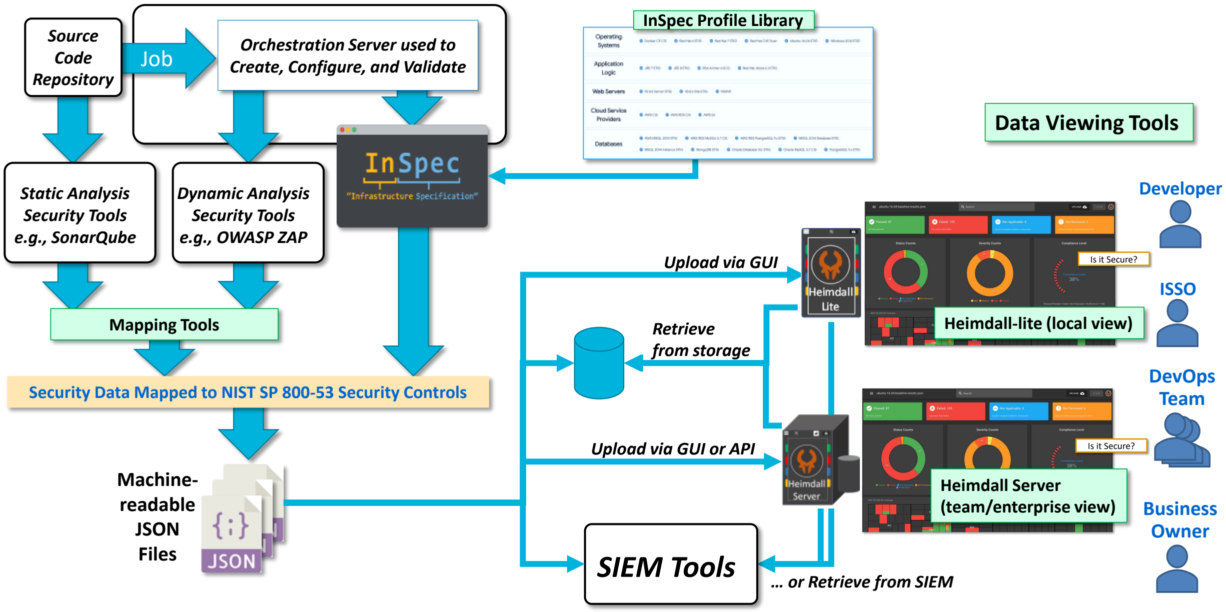

Figure 4-12 Data Viewing Tools – Heimdall + SIEM Tools (e.g., Splunk)

DevOps teams should make should use technologies to make this data as accessible as possible for different use cases and stakeholders. This will allow each stakeholder to analyze and address security defects and risks as soon as possible throughout the lifecycle of applications.

- For example, the open-source Heimdall-lite[10] provides functionality for viewing data from HDF JSON files. It is intended as a standalone single-file html page that can be loaded into a browser on a developer's own workstation. Files can be uploaded through the browser graphic user interface (GUI), from storage such as S3, or retrieved from SIEM sources such as Splunk.
- As another example, the open-source Heimdall Server[11] is a full standalone server version, providing storage for uploaded files, role-based access controls (RBAC), comparing of different JSON files, and trending of security debt over time. It is intended to be used as a local DevOps team server to analyze trends during Sprints. In addition to the upload methods demonstrated by Heimdall-lite, Heimdall Server allows upload via an API as well.

[10] https://heimdall-lite.cms.gov/
[11] https://github.com/mitre/heimdall2

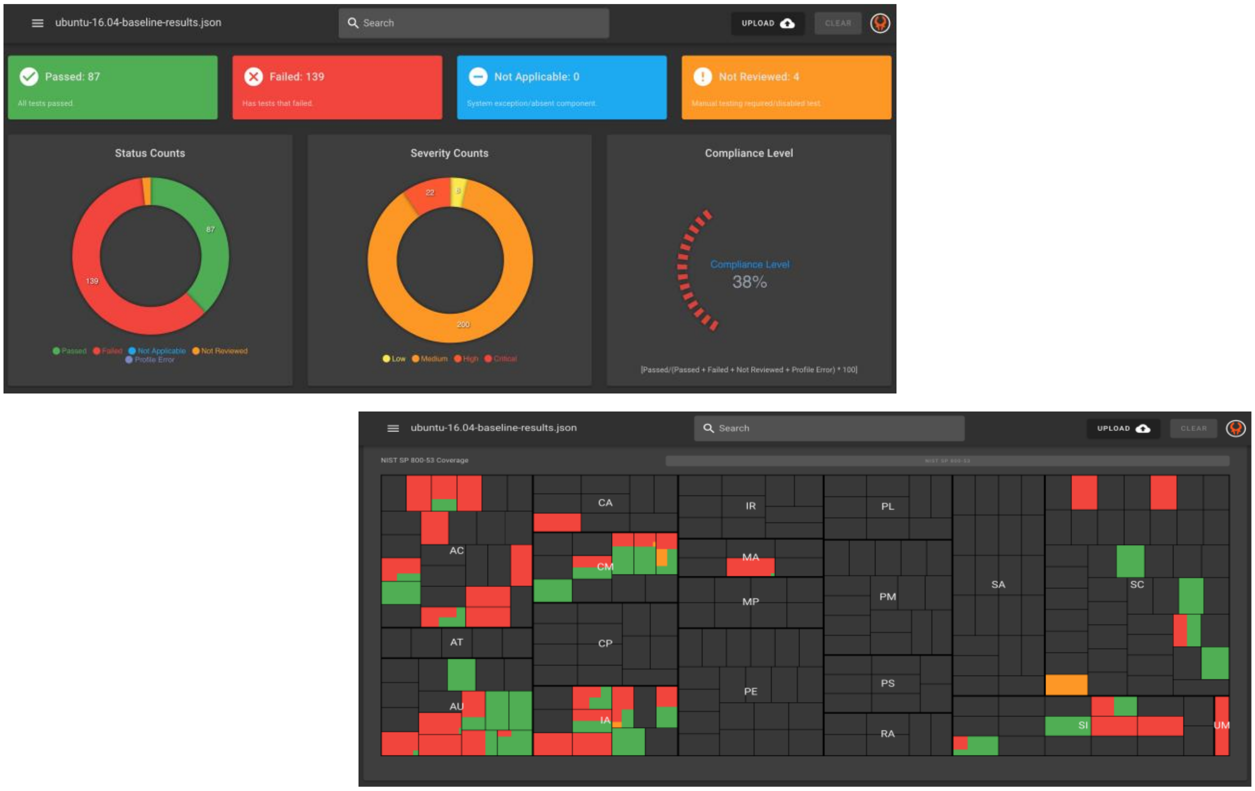

Figure 4-13 Data Viewing with Heimdall – Summary and Graphical Views

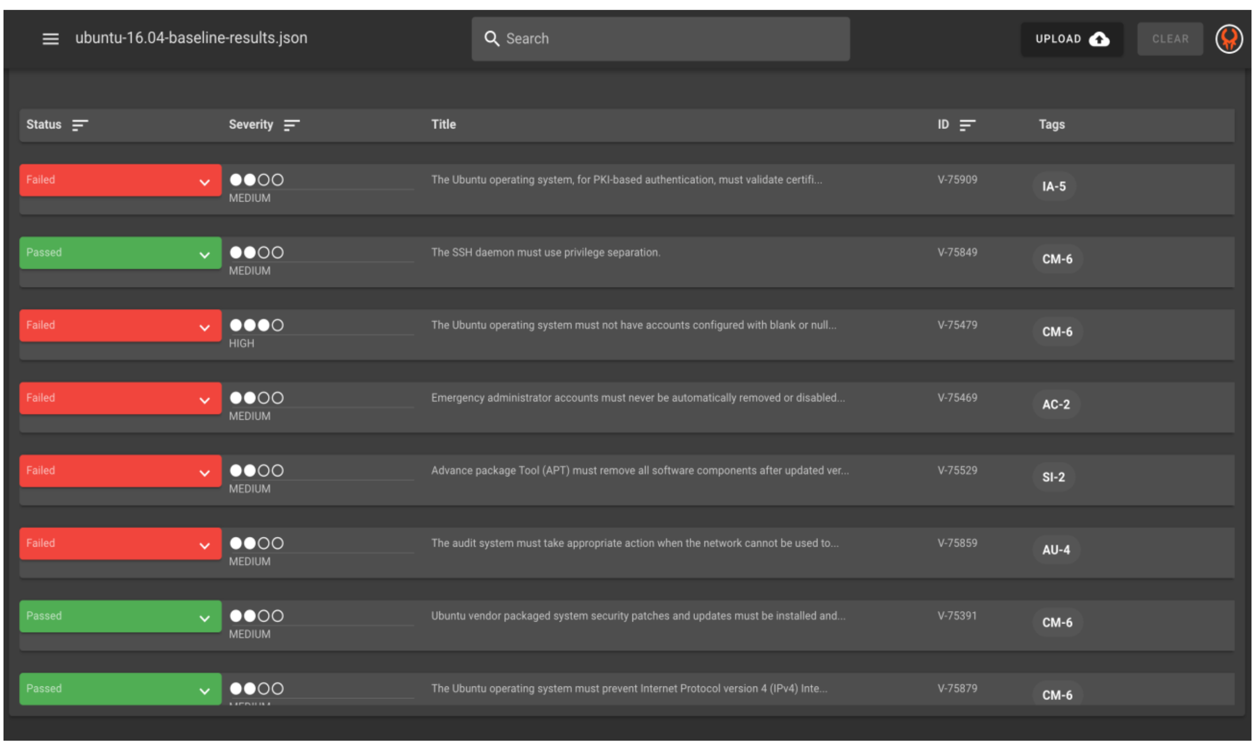

Figure 4-14 Example Data Viewing with Heimdall – Test Results List View

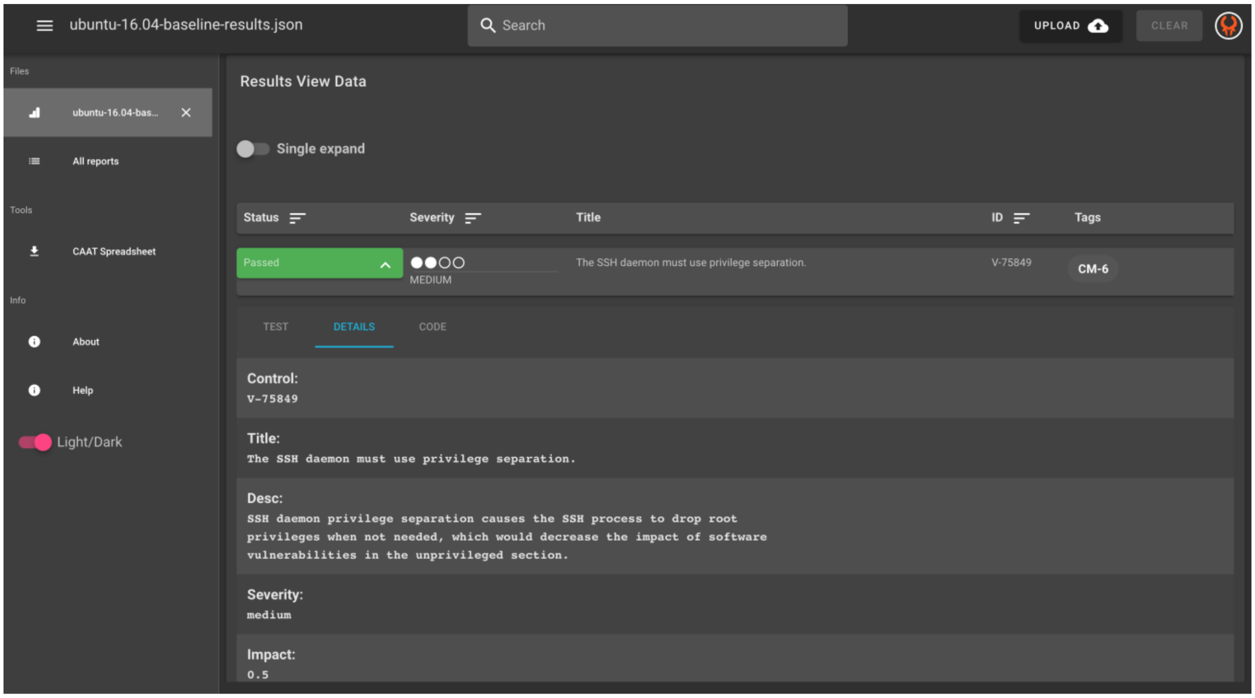

Figure 4-15 Example Data Viewing with Heimdall – Details View

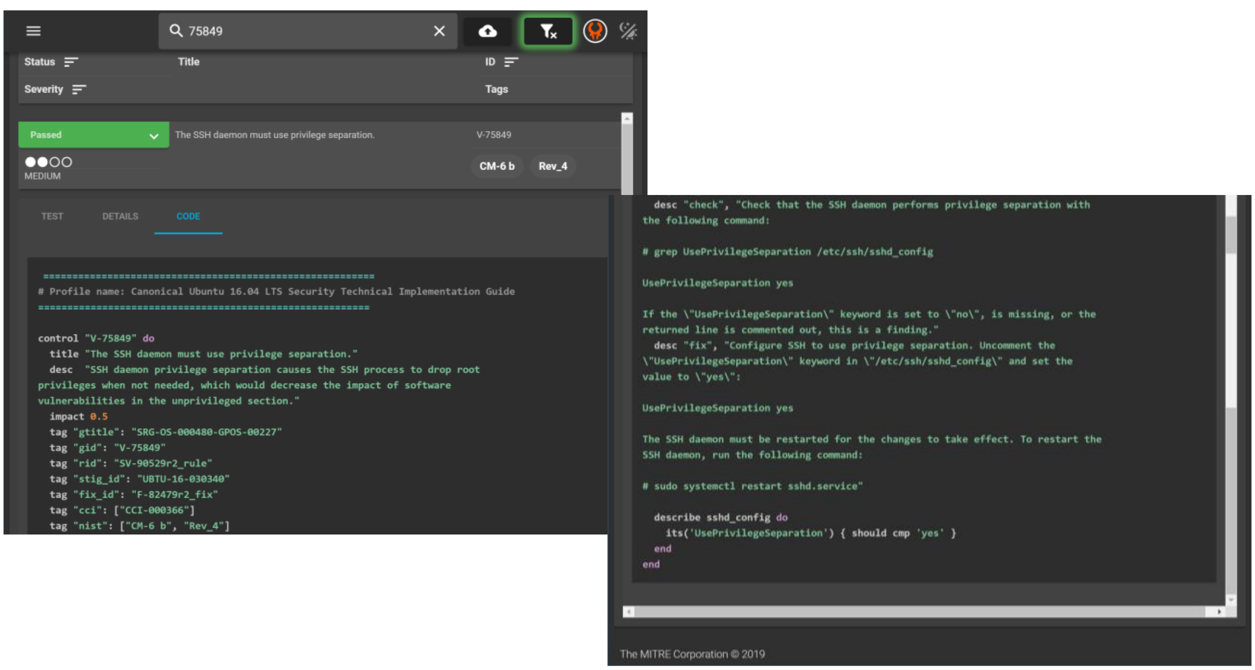

Figure 4-16 Example Data Viewing with Heimdall – InSpec Testing Code View

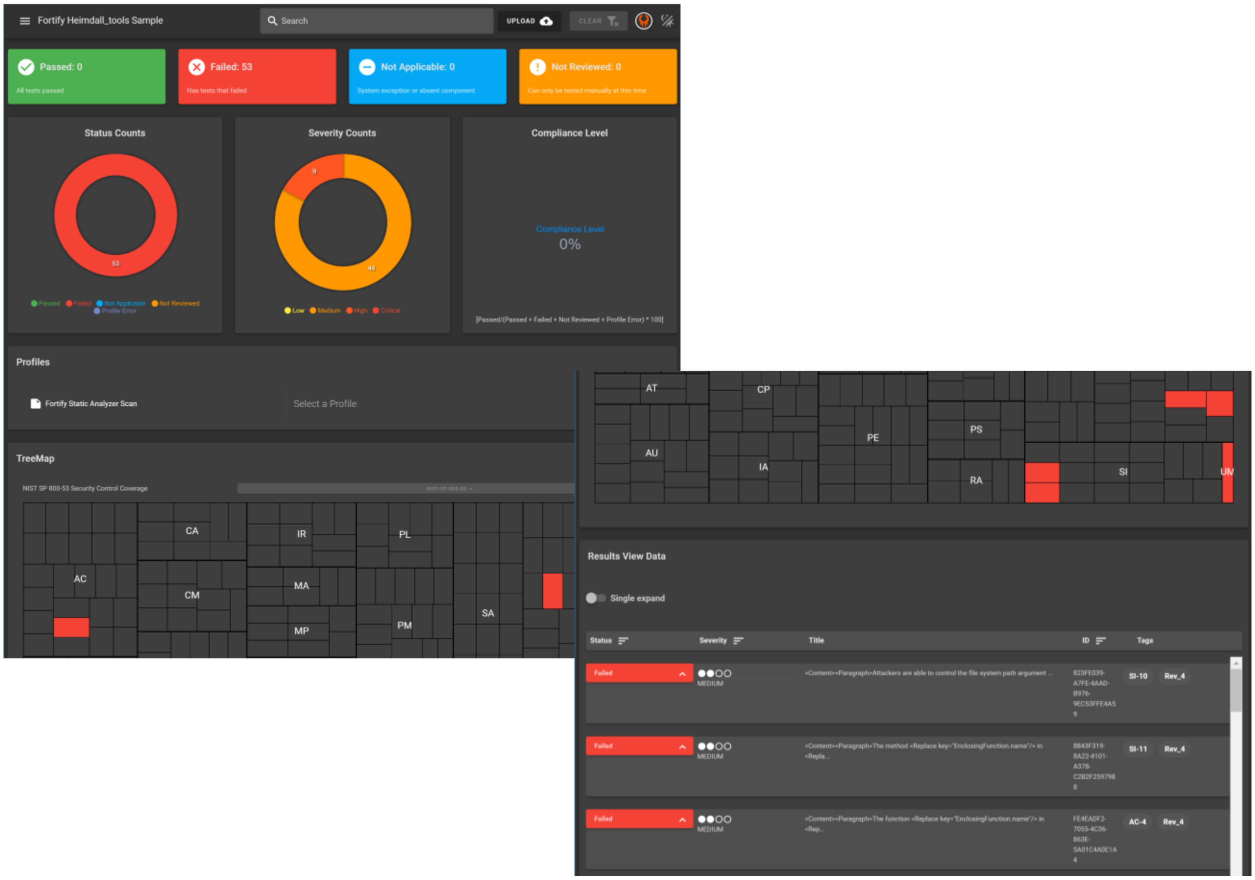

Figure 4-17 Example Data Viewing with Heimdall – Static Code Analysis Tool Output View

4.1.2 Security Documentation as Code Best Practice

The Security Documentation as Code focus area supports the user story for "How is it secure?":

"As a DevOps/Agile developer, maintaining security documentation is a challenge. In addition to validating security, I need tools that are integrated into my CI/CD pipeline to help me automatically document the as-built security in my system for my ISSO, Sprint-to-Sprint."

This focus area uses the pipeline to directly perform security validation tests. InSpec profiles include documentation for each security test they perform, and hence provide the context of each successful security check they perform. Each security check is also linked to a NIST SP 800-53 control. We can group this data by security control to help populate or supplement private implementation details in the SSP.

The Heimdall 1.0 viewer[12] provided a different view of the JSON data output for SSP-ready use:

[12] https://mitre.github.io/heimdall-lite-1.0/

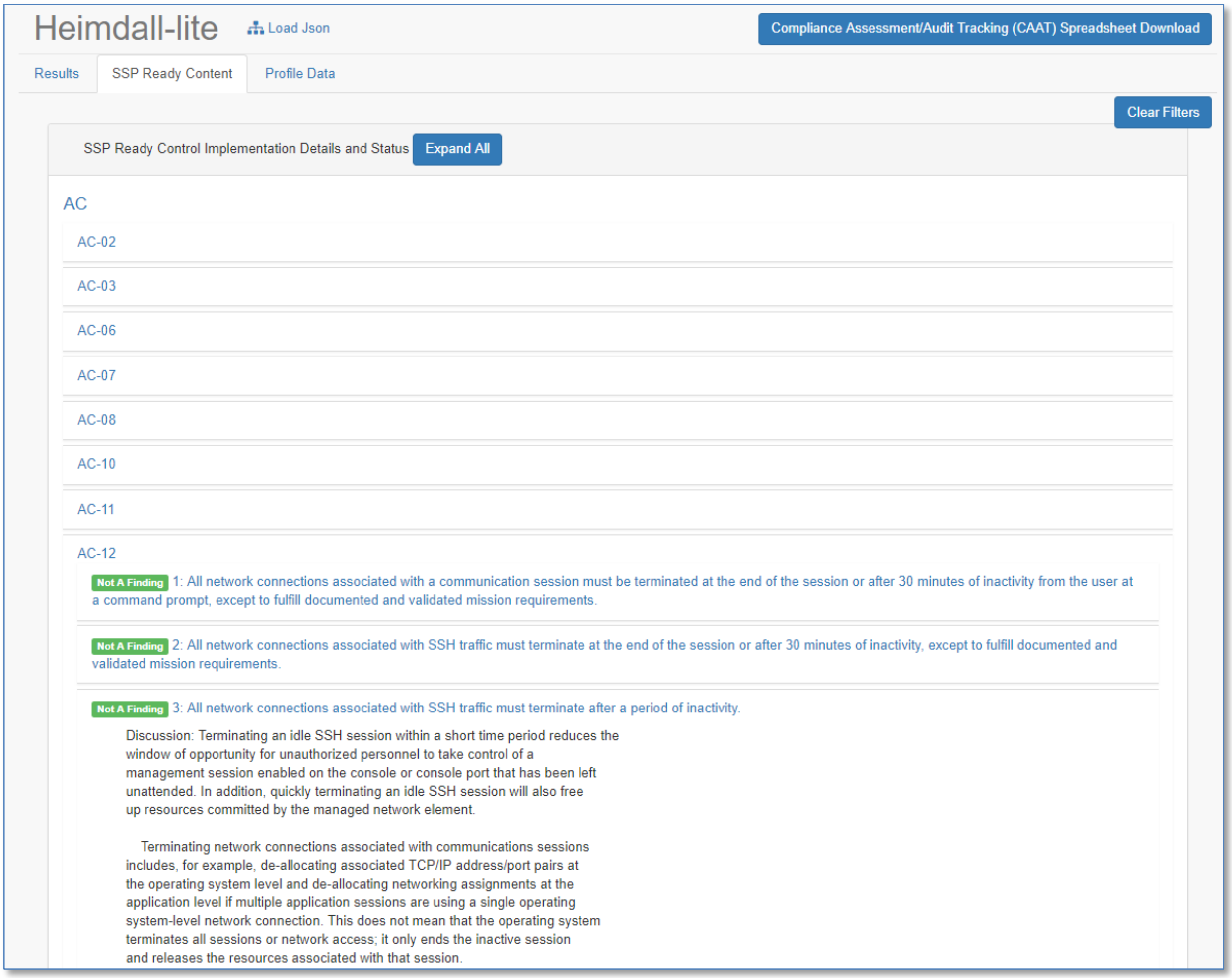

Figure 4-18 Using InSpec Profile output for Security Documentation

This view can be saved as an addendum for the ISSO to supplement the SSP documentation uploaded into CFACTS.

DevOps Teams should also incorporate the National Institutes of Standards and Technology (NIST) Open Security Controls Assessment Language (OSCAL)[13] standards into all aspects of security testing to support population and maintenance of SSP documentation:

> *"NIST, in collaboration with industry, is developing the Open Security Controls Assessment Language (OSCAL). OSCAL is a set of formats expressed in XML, JSON, and YAML. These formats provide machine-readable representations of control catalogs, control baselines, system security plans, and assessment plans and results."*

4.1.3 Change Management Auditing Best Practice

The Change Management Auditing focus area supports the user story for "What has changed during this Sprint?":

[13] https://pages.nist.gov/OSCAL/

"As a DevOps/Agile developer, I need to account for all my changes through the pipeline before requesting a security go-live decision for my system, Sprint-to-Sprint."

This focus area uses data from the pipeline itself, to correlate Jira issues to GitHub commits to orchestration server builds. Anything that can't be traced back to Jira is flagged as an unauthorized change for discussion with the ISSO during the Sprint.

At the beginning of each Sprint, the intent is for the ISSO to develop an initial Security Impact Analysis based on planned change for the Sprint, to foresee significant security testing changes in a Sprint. Near the end of the Sprint, this change management auditing best practice is intended to identify any unauthorized changes implemented during the Sprint.

CMS has chosen Splunk to support the collection of logs from pipeline components and to correlate the events across these components:

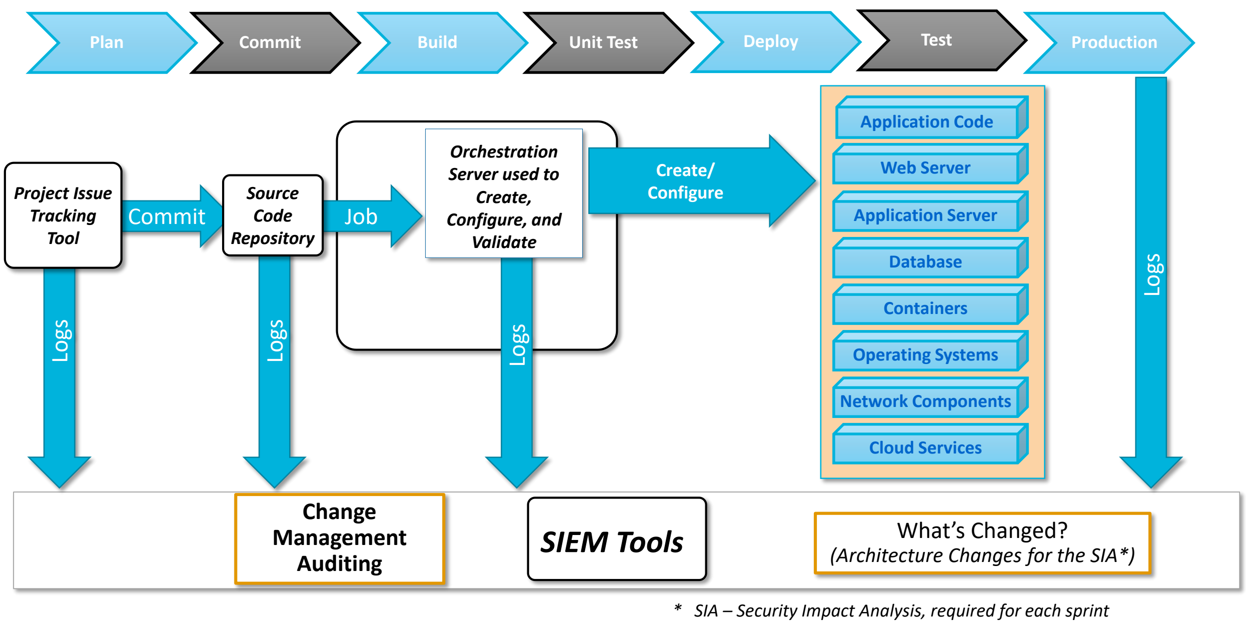

Figure 4-19 Change Management Auditing Best Practice Area supported by an SIEM (e.g., Splunk)

A dashboard could then be used to brings logs from each component of the pipeline together for the developer and ISSO to review.

In addition, the DevOps team should follow these code review recommendations:

Implement Project Code Standards for DevOps:

1. Employ code quality standards (for measuring code quality)
    a. Linting - used for checking for code clarity and neatness in code
    b. Code Coverage to 90-95% - unit/functional testing to exercise 90-95% of all functions
    c. Complexity Reduction - to increase maintainability and modularity, helps developers fix multiple instances of security problems with fewer corrections to code.
    d. Static Code review (security review)

        i. Seek a tool that attempts to check as many applicable Common Weaknesses Enumeration (CWE) types CWE, from SANS Top 25.
        ii. Note however that few tools reach much higher than a 30% true positive rate.[14]
        iii. Running tools from different sources can help identify true positives.
    e. Peer code review by team members (informal, analogous to independent release-based review)
2. Integrate all tools into the CI/CD pipeline – Automate to save time! Spend time evaluating the results rather than running the tool manually.
3. Code quality tools are selected as appropriate for the entire code base used for the project (e.g., node.js (JavaScript), ruby, etc.)[15]

Employ a Good Workflow:

1. Use branch/merge pull request model that rebases against the main line:
   → Upon every merge of a pull request into the main line, run code quality tools.
2. In the source code repository: All code commits must be tagged/related to a planned project issue ticket
3. In the project issue tracker: All planned project issue tickets for coding/recoding a function must be tagged with applicable security controls, with the help of the on-staff security engineer:
    a. Tag (identify) the security controls the function supports for the application - e.g., the application's authentication service directly supports (performs) AC-3 (Access Control Enforcement)
    b. Tag security controls that support the security of the function itself (e.g., the application's authentication service is protected by file permissions, encryption, employed with infrastructure components)
4. 1, 2, and 3 together provide a way to map pull requests to controls, to support security-based change control tracking.

4.2 Control Coverage for Current Best Practices Focus Areas

For Federal systems, relating any security defect or findings to a NIST SP 800-53 security control is vital to provide a common point of reference across all stakeholders making security decisions. The current best practices on security validation, documentation, and change management auditing validate or support the following security controls:

[14] https://rawgit.com/OWASP/Benchmark/master/scorecard/OWASP_Benchmark_Home.html
[15] SonarQube plugin for JavaScript exists - https://docs.sonarqube.org/display/PLUG/SonarJS

Security Validation as Code supports	**SA-11 – Developer Security Testing and Evaluation**
Security Documentation as Code supports	**SA-5 - Information System Documentation**

… for a **range of security controls** on Cloud (e.g., AWS), OS (e.g., Red Hat), Web (e.g., NGINX), or Database (e.g., PostgreSQL):

Access Control

Control Number	Cloud	OS	Web	DB
AC-02			✓	
AC-02(01)				✓
AC-02(04)	✓			
AC-03		✓	✓	✓
AC-06	✓		✓	
AC-06(07)	✓			
AC-06(09)	✓	✓		
AC-06(10)		✓		
AC-07		✓		
AC-08		✓		
AC-10		✓		✓
AC-11		✓		
AC-12		✓		✓
AC-16				✓
AC-17(02)		✓		
AC-18(01)		✓		

Audit and Accountability

Control Number	Cloud	OS	Web	DB
AU-02	✓			
AU-03		✓		✓
AU-03(01)				✓
AU-03(02)				✓
AU-04	✓	✓		✓
AU-05		✓		
AU-05(01)		✓		✓
AU-05(02)				✓
AU-06			✓	
AU-06(05)	✓			
AU-08				✓
AU-08(01)		✓		
AU-09	✓	✓		✓
AU-10				✓
AU-12	✓	✓		✓
AU-12(03)				✓

Configuration Management

Control Number	Cloud	OS	Web	DB
CM-02			✓	
CM-05(01)				✓
CM-06		✓	✓	
CM-07		✓	✓	✓
CM-07(01)				✓
CM-08	✓			
CM-08(02)	✓			
CM-08(03)	✓			

Identification and Authentication

Control Number	Cloud	OS	Web	DB
IA-02	✓	✓		✓
IA-02(01)	✓			
IA-02(02)		✓		
IA-02(11)		✓		
IA-02(12)		✓		
IA-03		✓		
IA-04	✓	✓		
IA-05			✓	
IA-05(01)	✓	✓		✓
IA-05(02)				✓
IA-07				✓
IA-08				✓

Incident Response

Control Number	Cloud	OS	Web	DB
IR-07	✓			

System and Communications Protection

Control Number	Cloud	OS	Web	DB
SC-02			✓	
SC-03			✓	✓
SC-04				✓
SC-05		✓		
SC-07	✓		✓	
SC-07(05)	✓			
SC-08		✓	✓	
SC-08(02)				✓
SC-10		✓		
SC-12	✓		✓	
SC-13				✓
SC-23	✓			✓
SC-28	✓			✓
SC-28(01)				✓

System and Information Integrity

Control Number	Cloud	OS	Web	DB
SI-02				✓
SI-03		✓		
SI-04(02)	✓			
SI-04(04)	✓			
SI-04(05)	✓			
SI-06		✓		
SI-07		✓		
SI-07(01)		✓		
SI-07(02)		✓		
SI-07(05)		✓		
SI-10				✓
SI-11				✓

Change Management Auditing supports
CM-3 - Configuration Change Control
SA-10 - Developer Configuration Management
CM-4 - Security Impact Analysis
CM-5(2) - Review System Changes
IR-6(1) - Automated (Incident) Reporting

Figure 4-20 NIST SP 800-53 Security Control Coverage when Building Security into DevOps

Note: This figure only covers 4 types of InSpec profiles. Appendix A details the control coverage for these 4 types of representative profiles. Appendices B and C map controls to key standards used by static and dynamic code analysis tools.

A maintained Control Range chart is maintained at https://saf.cms.gov/#/control-table

4.3 Future Best Practice Focus Areas

4.3.1 Reporting Best Practices

This focus area will develop best practice reporting and integration requirements to comply with stakeholder use of security data from the DevSecOps lifecycle. Stakeholders include developers, ISSOs, CRAs, ACT, CCIC, and CFACTS (FISMA reporting).

Developer and ISSO day-to-day use of security data produced by the DevOps pipeline has been covered by the current focus areas discussed in section 4.1.

4.3.1.1 ISSO Reporting

The ISSO is also responsible for tracking in CFACTS those security defects that cannot be resolved before the end of each Sprint. The standard method for populating any kind of security finding, be it from InSpec, static or dynamic code analysis tools, penetration testing, or external auditors, is to record these in a CMS Assessment/Audit Tracking (CAAT) spreadsheet for upload into CFACTS. Heimdall provides a means for the ISSO to generate this spreadsheet:

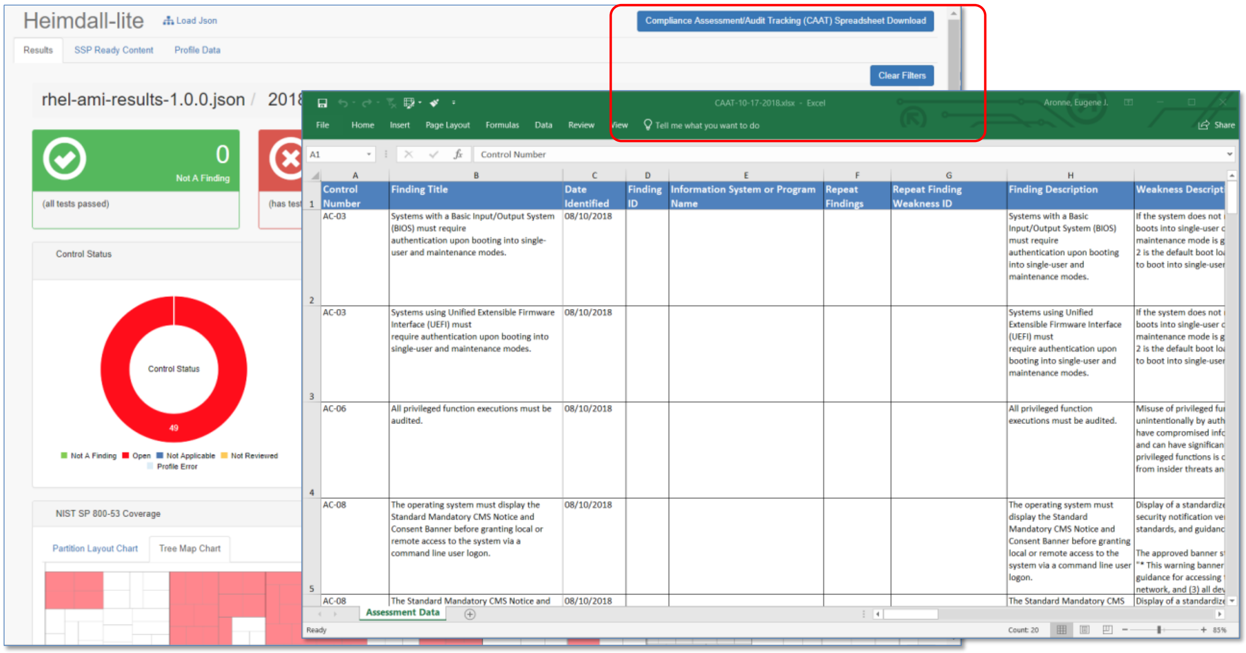

Figure 4-21 CMS Assessment/Audit Tracking (CAAT) Spreadsheet Generation using Heimdall

4.3.1.2 Cyber Risk Advisor Reporting

The Cyber Risk Advisors provide guidance to the ISSOs, helping ISSOs to prioritize remediation of security defects, especially challenging defects. Hence, the Cyber Risk Advisors require easy access to the same data viewed by the ISSOs. The proposed best practice to the CRA team is to provide access to the data via any of the Heimdall viewing options described in section 4.1.1.4, including viewing JSON files provided directly by the ISSO to their CRA.

4.3.1.3 ACT Reporting

The Adaptive Capabilities Testing team perform security assessments for the initial authorization of a system and ongoing authorization after the initial authorization:

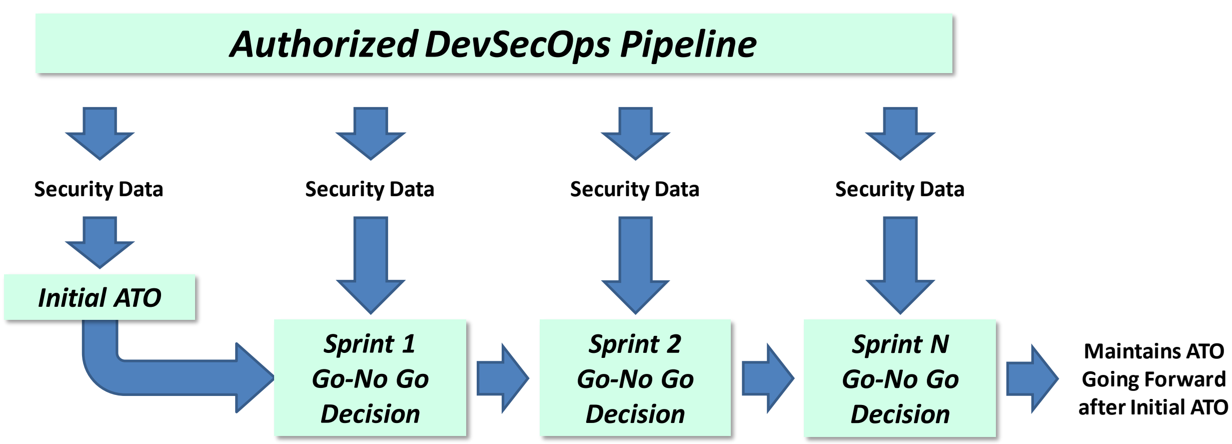

Figure 4-22 Adaptive Capabilities Testing Team use of DevSecOps Security Data for ATO Processes

The proposed best practice is for ACT to review security data from DevOps environments is by using Heimdall (discussed in 4.1.1.4) or CAAT (discussed in 4.2.1.1) options. This best practice will be refined through continued integration work between the ACT team and the ISPG DevSecOps research team.

4.3.1.4 CCIC Reporting

The CMS Cybersecurity Integration Center security operations team monitors security across the CMS enterprise, performing threat modeling and alerting system developers and maintainers when threats are detected targeting vulnerable CMS systems. CCIC needs to be aware of the security vulnerabilities of CMS systems. The proposed best practice for CCIC to review security data from DevOps environments is by using a Heimdall enterprise-level viewing capability provided through Splunk (discussed in 4.1.1.4). This best practice will be refined through continued integration work between CCIC and the ISPG DevSecOps research team.

4.3.1.5 CFACTS FISMA Tagging and Reporting

A common challenge across CMS security visibility efforts is the association of security data with its supporting CMS FISMA application/mission. This "FISMA Tagging" project is separate from DevSecOps work, striving to provide a standard approach to addressing this challenge. Hence, the ISPG DevSecOps research team will coordinate with the FISMA tagging team to ensure security data generated by the DevOps pipeline can be associated with the appropriate FISMA system.

4.3.2 Operational Analytics Best Practices

This focus area will develop a best practice process to engineer application audit log triggers during development to detect anomalies during operations and use this data to adapt to and plan for the next application development Sprint:

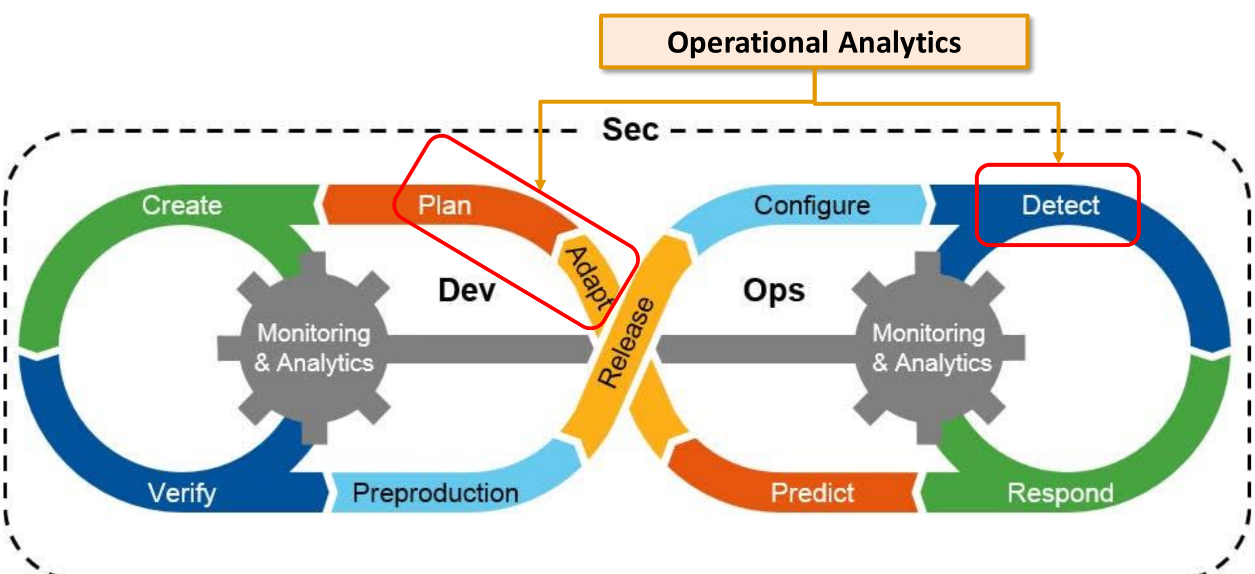

Figure 4-23 Operational Analytics Best Practice Support for DevSecOps "Infinity Loop"

The intent for this best practice is to work with the business owner and developers to define unusual business-application usage patterns to base security event triggers, leveraging web, API, and application server logs. Unusual usage patterns are highly dependent on the specific CMS business workflow and use cases for their application. These patterns could be unusual transactions, deletions, downloads of unusual volumes or types data given a user's defined role for the application, etc. Custom code may be required to generate many of these CMS business application-specific events. The intent is for Splunk to collect the audit logs and be configured to detect and alert the DevOps team of these anomalies.

4.3.3 DevSecOps Process Improvement Best Practices

This focus area will describe what to measure and how to analyze the data to constantly improve the project's DevSecOps process. It will improve future builds using metrics and measures of security debt, unauthorized changes during development, and detection of anomalies during operation:

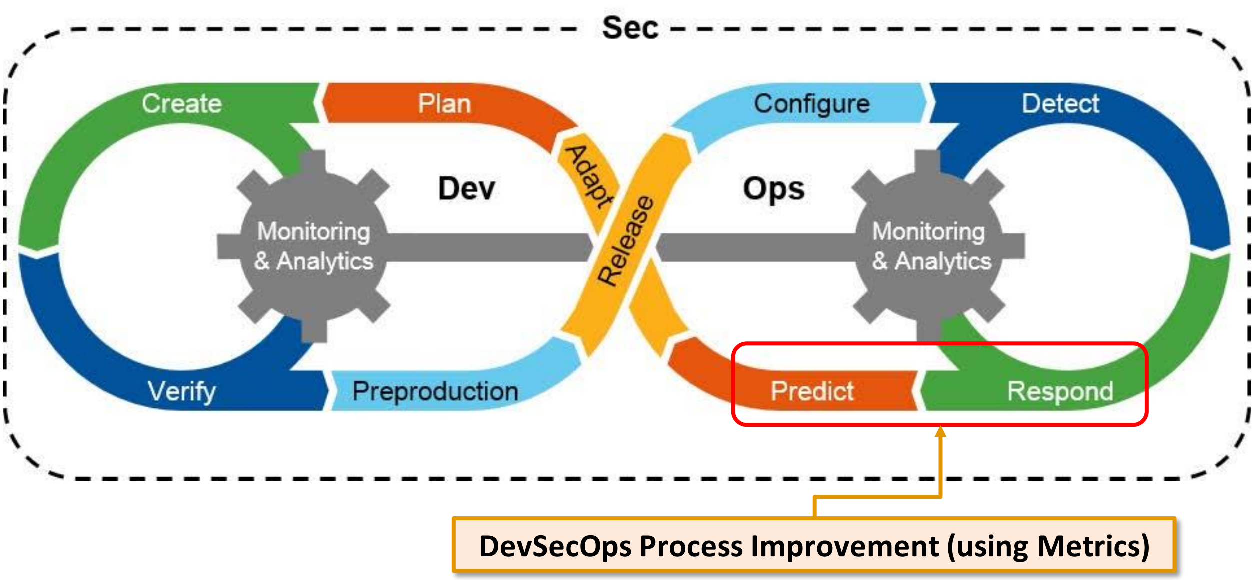

Figure 4-24 DevSecOps Cycle Improvement Best Practice Support for DevSecOps "Infinity Loop"

Initial proposed metrics include:

- All high security defects and 90% of all medium or low security defects are resolved before allowing affected functionality to be deployed to production.
- No security defect may carry-over unresolved through more than 2 Sprints.
- The number of unplanned (unauthorized) changes for any Sprint is less than 5% of the number of planned (authorized) changes.

This best practice should be refined through continued work with internal and external DevOps teams, including industry best practice research such as the DORA (DevOps Research and Assessment)[16].

[16] https://devops-research.com/

Appendix A. NIST SP 800-53 Security Control Coverage Details

CMS ARS 3.1 Control Coverage WorkSheet | **Assessment Target:** *CIS AWS Foundations Benchmark v1.1.0 - 11-29-2016* | **Date:** *3/1/2018* | **Comment:** *InSpec Profile*

Control Family		Control Number(s) (* indicates a non-mandatory control)																												
Access Control	AC	1	2	2(1)	2(2)	2(3)	2(4)	2(5)	2(9)*	2(10)*	2(11)	2(12)	2(13)	3	3(9)*	4	4(8)*	4(12)*	4(15)*	4(17)*	4(18)*	5	6	6(1)	6(2)	6(3)	6(5)	6(7)*	6(9)	6(10)
		7	8	9*	9(1)*	10	11	11(1)	12	14	16*	16(3)*	17	17(1)	17(2)	17(3)	17(4)	17(9)	18	18(1)	18(4)	18(5)	19	19(5)	20	20(1)	20(2)	20(3)*	21	22
Awareness and Training	AT	1	2	2(2)	3	4																								
Audit and Accountability	AU	1	2	2(3)	3	3(1)	3(2)	4	5	5(1)	5(2)	6	6(1)	6(3)	6(5)	6(6)	6(10)*	7	7(1)	7(2)*										
		8	8(1)	9	9(2)	9(3)	9(4)	10	10(1)*	11	12	12(1)	12(3)	16*	16(2)*															
Configuration Management	CM	1	2	2(1)	2(2)	2(3)	2(6)*	2(7)	3	3(1)	3(2)	3(6)*	4	4(1)	4(2)*	5	5(1)	5(2)	5(3)											
		6	6(1)	6(2)	7	7(1)	7(2)	7(4)	7(5)*	7(5)	8	8(1)	8(2)	8(3)	8(4)	8(5)	9	10	11											
Contingency Planning	CP	7	7(1)	7(2)	7(3)	7(4)	8	8(1)	8(2)	8(3)	8(4)	9	9(1)	9(2)	9(3)	9(5)	10	10(2)	10(4)											
Identification and Authentication	IA	1	2	2(1)	2(2)	2(3)	2(4)	2(6)*	2(7)*	2(8)	2(9)	2(11)	2(12)	3	4	4(3)*	5	5(1)	5(2)	5(3)	5(11)	6	7	8	8(1)	8(2)	8(3)	8(4)		
Incident Response	IR	1	2	2(1)	2(2)	3	3(1)*	3(2)	4	4(1)	4(3)*	4(4)	4(6)*	4(7)*	5	5(1)	6	6(1)	7	7(1)	7(2)*	8	9*	9(1)*	9(2)*	9(3)*	9(4)*	10*		
Maintenance	MA	1	2	2(2)	3	3(1)	3(2)	3(3)	4	4(1)	4(2)	4(3)	4(6)*	5	5(1)	6														
Media Protection	MP	1	2	3	4	5	5(3)*	5(4)	6	6(1)	6(2)	6(3)	6(8)*	7	7(1)	8*	8(3)*	C-1												
Physical and Environmental Protection	PE	1	2	2(1)*	3	3(1)	4	5	6	6(1)	6(4)	8	8(1)	9	10	11	11(1)	12	13	13(1)	13(2)	13(3)	14	15	15(1)	16	17	18	18(1)*	
Planning	PL	1	2	2(3)	4	4(1)	8																							
Personnel Security	PS	1	2	3	3(3)*	4	4(2)	5	6	7	8																			
Risk Assessment	RA	1	2	3	5	5(1)	5(2)	5(3)*	5(4)	5(5)																				
System and Services Acquisition	SA	1	2	3	4	4(1)	4(2)	4(8)*	4(9)	4(10)	5	8	9	9(2)	9(5)*	10	10(1)*	11	11(2)*	11(5)*	11(8)*	12	13*	15	15(9)	16	17	21*	22*	
System and Communications Protection	SC	1	2	3	3(2)*	3(3)*	3(4)*	3(5)*	4	5	7	7(3)	7(4)	7(5)	7(7)	7(8)	7(14)*	7(18)	7(21)	8	8(1)	8(2)*								
		10	12	12(1)	13	15	15(1)	17	18	19	20	21	22	23	24	28	28(1)*	32*	39	C-1	C-2									
System and Information Integrity	SI	1	2	2(1)	2(2)	2(3)*	3	3(1)	3(2)	4	4(1)*	4(2)	4(3)*	4(4)	4(5)	4(14)*	4(16)	4(23)*	4(24)*											
		5	5(1)	6	6(2)*	7	7(1)	7(2)	7(5)	7(6)*	7(7)	7(14)	8	8(1)	8(2)	10	11	12	16											

Assessment Legend:

Green indicates a covered control for the assessment target

Figure 4-25 AWS InSpec Profile - Security Control Coverage

CMS ARS 3.1 Control Coverage WorkSheet	Assessment Target:	*Red Hat Enterprise Linux 7 STIG R3 27 Oct 2017*	Date:	*3/1/2018*	Comment:	*InSpec Profile*

Control Family		Control Number(s) *(* indicates a non-mandatory control)*																												
Access Control	AC	1	2	2(1)	2(2)	2(3)	2(4)	2(5)	2(9)*	2(10)*	2(11)	2(12)	2(13)	3	3(9)*	4	4(8)*	4(12)*	4(15)*	4(17)*	4(18)*	5	6	6(1)	6(2)	6(3)	6(5)	6(7)*	6(9)	6(10)
		7	8	9*	9(1)*	10	11	11(1)	12	14	16*	16(3)*	17	17(1)	17(2)	17(3)	17(4)	17(9)	18	18(1)	18(4)	18(5)	19	19(5)	20	20(1)	20(2)	20(3)*	21	22
Awareness and Training	AT	1	2	2(2)	3	4
Audit and Accountability	AU	1	2	2(3)	3	3(1)	3(2)	4	5	5(1)	5(2)	6	6(1)	6(3)	6(5)	6(6)	6(10)*	7	7(1)	7(2)*
		8	8(1)	9	9(2)	9(3)	9(4)	10	10(1)*	11	12	12(1)	12(3)	16*	16(2)*
Configuration Management	CM	1	2	2(1)	2(2)	2(3)	2(6)*	2(7)	3	3(1)	3(2)	3(6)*	4	4(1)	4(2)*	5	5(1)	5(2)	5(3)
		6	6(1)	6(2)	7	7(1)	7(2)	7(4)	7(5)*	7(5)	8	8(1)	8(2)	8(3)	8(4)	8(5)	9	10	11
Contingency Planning	CP	7	7(1)	7(2)	7(3)	7(4)	8	8(1)	8(2)	8(3)	8(4)	9	9(1)	9(2)	9(3)	9(5)	10	10(2)	10(4)
Identification and Authentication	IA	1	2	2(1)	2(2)	2(3)	2(4)	2(6)*	2(7)*	2(8)	2(9)	2(11)	2(12)	3	4	4(3)*	5	5(1)	5(2)	5(3)	5(11)	6	7	8	8(1)	8(2)	8(3)	8(4)
Incident Response	IR	1	2	2(1)	2(2)	3	3(1)*	3(2)	4	4(1)	4(3)*	4(4)	4(6)*	4(7)*	5	5(1)	6	6(1)	7	7(1)	7(2)*	8	9*	9(1)*	9(2)*	9(3)*	9(4)*	10*
Maintenance	MA	1	2	2(2)	3	3(1)	3(2)	3(3)	4	4(1)	4(2)	4(3)	4(6)*	5	5(1)	6
Media Protection	MP	1	2	3	4	5	5(3)*	5(4)	6	6(1)	6(2)	6(3)	6(8)*	7	7(1)	8*	8(3)*	C-1
Physical and Environmental Protection	PE	1	2	2(1)*	3	3(1)	4	5	6	6(1)	6(4)	8	8(1)	9	10	11	11(1)	12	13	13(1)	13(2)	13(3)	14	15	15(1)	16	17	18	18(1)*
Planning	PL	1	2	2(3)	4	4(1)	8
Personnel Security	PS	1	2	3	3(3)*	4	4(2)	5	6	7	8
Risk Assessment	RA	1	2	3	5	5(1)	5(2)	5(3)*	5(4)	5(5)
System and Services Acquisition	SA	1	2	3	4	4(1)	4(2)	4(8)*	4(9)	4(10)	5	8	9	9(2)	9(5)*	10	10(1)*	11	11(2)*	11(5)*	11(8)*	12	13*	15	15(9)	16	17	21*	22*
System and Communications Protection	SC	1	2	3	3(2)*	3(3)*	3(4)*	3(5)*	4	5	7	7(3)	7(4)	7(5)	7(7)	7(8)	7(14)*	7(18)	7(21)	8	8(1)	8(2)*
		10	12	12(1)	13	15	15(1)	17	18	19	20	21	22	23	24	28	28(1)*	32*	39	C-1	C-2
System and Information Integrity	SI	1	2	2(1)	2(2)	2(3)*	3	3(1)	3(2)	4	4(1)*	4(2)	4(3)*	4(4)	4(5)	4(14)*	4(16)	4(23)*	4(24)*
		5	5(1)	6	6(2)*	7	7(1)	7(2)	7(5)	7(6)*	7(7)	7(14)	8	8(1)	8(2)	10	11	12	16

Assessment Legend:

Green indicates a covered control for the assessment target

Figure 4-26 Red Hat InSpec Profile – Security Control Coverage

CMS ARS 3.1 Control Coverage WorkSheet	Assessment Target:	NGINX Web Server Config (based on Apache STIG 2.2)	Date:	3/1/2018	Comment:	InSpec Profile

Control Family		Control Number(s) (* indicates a non-mandatory control)																												
Access Control	AC	1	2	2(1)	2(2)	2(3)	2(4)	2(5)	2(9)*	2(10)*	2(11)	2(12)	2(13)	3	3(9)*	4	4(8)*	4(12)*	4(15)*	4(17)*	4(18)*	5	6	6(1)	6(2)	6(3)	6(5)	6(7)*	6(9)	6(10)
		7	8	9*	9(1)*	10	11	11(1)	12	14	16*	16(3)*	17	17(1)	17(2)	17(3)	17(4)	17(9)	18	18(1)	18(4)	18(5)	19	19(5)	20	20(1)	20(2)	20(3)*	21	22
Awareness and Training	AT	1	2	2(2)	3	4																								
Audit and Accountability	AU	1	2	2(3)	3	3(1)	3(2)	4	5	5(1)	5(2)	6	6(1)	6(3)	6(5)	6(6)	6(10)*	7	7(1)	7(2)*										
		8	8(1)	9	9(2)	9(3)	9(4)	10	10(1)*	11	12	12(1)	12(3)	16*	16(2)*															
Configuration Management	CM	1	2	2(1)	2(2)	2(3)	2(6)*	2(7)	3	3(1)	3(2)	3(6)*	4	4(1)	4(2)*	5	5(1)	5(2)	5(3)											
		6	6(1)	6(2)	7	7(1)	7(2)	7(4)	7(5)*	7(5)	8	8(1)	8(2)	8(3)	8(4)	8(5)	9	10	11											
Contingency Planning	CP	7	7(1)	7(2)	7(3)	7(4)	8	8(1)	8(2)	8(3)	8(4)	9	9(1)	9(2)	9(3)	9(5)	10	10(2)	10(4)											
Identification and Authentication	IA	1	2	2(1)	2(2)	2(3)	2(4)	2(6)*	2(7)*	2(8)	2(9)	2(11)	2(12)	3	4	4(3)*	5	5(1)	5(2)	5(3)	5(11)	6	7	8	8(1)	8(2)	8(3)	8(4)		
Incident Response	IR	1	2	2(1)	2(2)	3	3(1)*	3(2)	4	4(1)	4(3)*	4(4)	4(6)*	4(7)*	5	5(1)	6	6(1)	7	7(1)	7(2)*	8	9*	9(1)*	9(2)*	9(3)*	9(4)*	10*		
Maintenance	MA	1	2	2(2)	3	3(1)	3(2)	3(3)	4	4(1)	4(2)	4(3)	4(6)*	5	5(1)	6														
Media Protection	MP	1	2	3	4	5	5(3)*	5(4)	6	6(1)	6(2)	6(3)	6(8)*	7	7(1)	8*	8(3)*	C-1												
Physical and Environmental Protection	PE	1	2	2(1)*	3	3(1)	4	5	6	6(1)	6(4)	8	8(1)	9	10	11	11(1)	12	13	13(1)	13(2)	13(3)	14	15	15(1)	16	17	18	18(1)*	
Planning	PL	1	2	2(3)	4	4(1)	8																							
Personnel Security	PS	1	2	3	3(3)*	4	4(2)	5	6	7	8																			
Risk Assessment	RA	1	2	3	5	5(1)	5(2)	5(3)*	5(4)	5(5)																				
System and Services Acquisition	SA	1	2	3	4	4(1)	4(2)	4(8)*	4(9)	4(10)	5	8	9	9(2)	9(5)*	10	10(1)*	11	11(2)*	11(5)*	11(8)*	12	13*	15	15(9)	16	17	21*	22*	
System and Communications Protection	SC	1	2	3	3(2)*	3(3)*	3(4)*	3(5)*	4	5	7	7(3)	7(4)	7(5)	7(7)	7(8)	7(14)*	7(18)	7(21)	8	8(1)	8(2)*								
		10	12	12(1)	13	15	15(1)	17	18	19	20	21	22	23	24	28	28(1)*	32*	39	C-1	C-2									
System and Information Integrity	SI	1	2	2(1)	2(2)	2(3)*	3	3(1)	3(2)	4	4(1)*	4(2)	4(3)*	4(4)	4(5)	4(14)*	4(16)	4(23)*	4(24)*											
		5	5(1)	6	6(2)*	7	7(1)	7(2)	7(5)	7(6)*	7(7)	7(14)	8	8(1)	8(2)	10	11	12	16											

Assessment Legend:

Green indicates a covered control for the assessment target

Figure 4-27 NGINX InSpec Profile – Security Control Coverage

CMS ARS 3.1 Control Coverage WorkSheet	Assessment Target:	*PostgreSQL 9.x STIG*	Date:	*3/1/2018*	Comment:	*InSpec Profile*

Control Family		Control Number(s) (** indicates a non-mandatory control*)
Access Control	AC	1, 2, 2(1), 2(2), 2(3), 2(4), 2(5), 2(9)*, 2(10)*, 2(11), 2(12), 2(13), 3, 3(9)*, 4, 4(8)*, 4(12)*, 4(15)*, 4(17)*, 4(18)*, 5, 6, 6(1), 6(2), 6(3), 6(5), 6(7)*, 6(9), 6(10)
		7, 8, 9*, 9(1)*, 10, 11, 11(1), 12, 14, 16*, 16(3)*, 17, 17(1), 17(2), 17(3), 17(4), 17(9), 18, 18(1), 18(4), 18(5), 19, 19(5), 20, 20(1), 20(2), 20(3)*, 21, 22
Awareness and Training	AT	1, 2, 2(2), 3, 4
Audit and Accountability	AU	1, 2, 2(3), 3, 3(1), 3(2), 4, 5, 5(1), 5(2), 6, 6(1), 6(3), 6(5), 6(6), 6(10)*, 7, 7(1), 7(2)*
		8, 8(1), 9, 9(2), 9(3), 9(4), 10, 10(1)*, 11, 12, 12(1), 12(3), 16*, 16(2)*
Configuration Management	CM	1, 2, 2(1), 2(2), 2(3), 2(6)*, 2(7), 3, 3(1), 3(2), 3(6)*, 4, 4(1), 4(2)*, 5, 5(1), 5(2), 5(3)
		6, 6(1), 6(2), 7, 7(1), 7(2), 7(4), 7(5)*, 7(5), 8, 8(1), 8(2), 8(3), 8(4), 8(5), 9, 10, 11
Contingency Planning	CP	7, 7(1), 7(2), 7(3), 7(4), 8, 8(1), 8(2), 8(3), 8(4), 9, 9(1), 9(2), 9(3), 9(5), 10, 10(2), 10(4)
Identification and Authentication	IA	1, 2, 2(1), 2(2), 2(3), 2(4), 2(6)*, 2(7)*, 2(8), 2(9), 2(11), 2(12), 3, 4, 4(3)*, 5, 5(1), 5(2), 5(3), 5(11), 6, 7, 8, 8(1), 8(2), 8(3), 8(4)
Incident Response	IR	1, 2, 2(1), 2(2), 3, 3(1)*, 3(2), 4, 4(1), 4(3)*, 4(4), 4(6)*, 4(7)*, 5, 5(1), 6, 6(1), 7, 7(1), 7(2)*, 8, 9*, 9(1)*, 9(2)*, 9(3)*, 9(4)*, 10*
Maintenance	MA	1, 2, 2(2), 3, 3(1), 3(2), 3(3), 4, 4(1), 4(2), 4(3), 4(6)*, 5, 5(1), 6
Media Protection	MP	1, 2, 3, 4, 5, 5(3)*, 5(4), 6, 6(1), 6(2), 6(3), 6(8)*, 7, 7(1), 8*, 8(3)*, C-1
Physical and Environmental Protection	PE	1, 2, 2(1)*, 3, 3(1), 4, 5, 6, 6(1), 6(4), 8, 8(1), 9, 10, 11, 11(1), 12, 13, 13(1), 13(2), 13(3), 14, 15, 15(1), 16, 17, 18, 18(1)*
Planning	PL	1, 2, 2(3), 4, 4(1), 8
Personnel Security	PS	1, 2, 3, 3(3)*, 4, 4(2), 5, 6, 7, 8
Risk Assessment	RA	1, 2, 3, 5, 5(1), 5(2), 5(3)*, 5(4), 5(5)
System and Services Acquisition	SA	1, 2, 3, 4, 4(1), 4(2), 4(8)*, 4(9), 4(10), 5, 8, 9, 9(2), 9(5)*, 10, 10(1)*, 11, 11(2)*, 11(5)*, 11(8)*, 12, 13*, 15, 15(9), 16, 17, 21*, 22*
System and Communications Protection	SC	1, 2, 3, 3(2)*, 3(3)*, 3(4)*, 3(5)*, 4, 5, 7, 7(3), 7(4), 7(5), 7(7), 7(8), 7(14)*, 7(18), 7(21), 8, 8(1), 8(2)*
		10, 12, 12(1), 13, 15, 15(1), 17, 18, 19, 20, 21, 22, 23, 24, 28, 28(1)*, 32*, 39, C-1, C-2
System and Information Integrity	SI	1, 2, 2(1), 2(2), 2(3)*, 3, 3(1), 3(2), 4, 4(1)*, 4(2), 4(3)*, 4(4), 4(5), 4(14)*, 4(16), 4(23)*, 4(24)*
		5, 5(1), 6, 6(2)*, 7, 7(1), 7(2), 7(5), 7(6)*, 7(7), 7(14), 8, 8(1), 8(2), 10, 11, 12, 16

Assessment Legend:

Green indicates a covered control for the assessment target

Figure 4-28 PostgreSQL InSpec Profile – Security Control Coverage

Appendix B. Security Control Mapping to SANS Top 25 CWE

Minimum CWE used by code analysis tools to categorize security defects (based on SANS Top 25 2011)	**Primary** NIST SP 800-53 Controls
CWE-89 Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	SI-10 - Information Input Validation
CWE-78 Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	**SI-10 - Information Input Validation** SI-3 - Malicious Code Protection
CWE-120 Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')	SI-16 - Memory Protection
CWE-79 Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	SI-10 - Information Input Validation
CWE-306 Missing Authentication for Critical Function	**IA-5 - Authenticator Management** AC-14 – Permitted Actions Without Identification or Authentication
CWE-862 Missing Authorization	AC-6 - Least Privilege
CWE-798 Use of Hard-coded Credentials	IA-2 - Identification and Authentication (Organizational Users)
CWE-311 Missing Encryption of Sensitive Data	**SC-13 - Cryptographic Protection** SC-28 - Protection of Information at rest
CWE-434 Unrestricted Upload of File with Dangerous Type	SI-10 - Information Input Validation
CWE-807 Reliance on Untrusted Inputs in a Security Decision	SI-10 - Information Input Validation
CWE-250 Execution with Unnecessary Privileges	AC-6 - Least Privilege
CWE-352 Cross-Site Request Forgery (CSRF)	**SC-23 - Session Authenticity** AC-10 – Concurrent Session Control AC-11 – Session Lock AC-14 – Permitted Actions Without Identification or Authentication
CWE-22 Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	**AC-3 – Access Enforcement** SI-10 – Information Input Validation
CWE-494 Download of Code Without Integrity Check	SI-7 - Software, Firmware, and Information Integrity
CWE-863 Incorrect Authorization	AC-3 - Access Enforcement
CWE-829 Inclusion of Functionality from Untrusted Control Sphere	SI-10 - Information Input Validation
CWE-732 Incorrect Permission Assignment for Critical Resource	AC-3 - Access Enforcement
CWE-676 Use of Potentially Dangerous Function	**SI-10 - Information Input Validation** SI-11 - Error Handling
CWE-327 Use of a Broken or Risky Cryptographic Algorithm	**SC-13 - Cryptographic Protection** SC-28 - Protection of Information at rest
CWE-131 Incorrect Calculation of Buffer Size	SI-10 - Information Input Validation
CWE-307 Improper Restriction of Excessive Authentication Attempts	**SC-23 - Session Authenticity** AC-10 – Concurrent Session Control AC-11 – Session Lock AC-14 – Permitted Actions Without Identification or Authentication
CWE-601 URL Redirection to Untrusted Site ('Open Redirect')	**SI-10 - Information Input Validation** AC-3 - Access Enforcement
CWE-134 Uncontrolled Format String	SI-10 - Information Input Validation
CWE-190 Integer Overflow or Wraparound	SI-10 - Information Input Validation
CWE-759 Use of a One-Way Hash without a Salt	**SC-13 - Cryptographic Protection** SC-12 - Cryptographic Key Establishment and Management

Appendix C. Security Control Mapping to OWASP Top 10

OWASP categories used by code analysis tools to categorize security defects (based on OWASP Top 10 2013)	**Primary** NIST SP 800-53 Controls
A1 Injection	SI-10 – Information Input Validation
A2 Broken Authentication and Session Management	**SC-23 - Session Authenticity** AC-10 – Concurrent Session Control AC-11 – Session Lock AC-14 – Permitted Actions Without Identification or Authentication
A3 Cross-Site Scripting (XSS)	SI-10 – Information Input Validation
A4 Insecure Direct Object References	**AC-3 – Access Enforcement** SI-10 – Information Input Validation
A5 Security Misconfiguration	CM-6 – Configuration Settings
A6 Sensitive Data Exposure	**SI-11 – Error Handling** SC-4 - Information in Shared Resources SC-28 – Protection of Information at Rest
A7 Missing Function Level Access Control	**CM-7 Least Functionality** AC-3 – Access Enforcement
A8 Cross-Site Request Forgery (CSRF)	SC-23 – Session Authenticity
A9 Using Components with Known Vulnerabilities	SI-2 – Flaw Remediation
A10 Unvalidated Redirects and Forwards	SI-10 – Information Input Validation

OWASP categories used by code analysis tools to categorize security defects (based on OWASP Top 10 2017)	**Primary** NIST SP 800-53 Controls
A1:2017-Injection	SI-10 – Information Input Validation
A2:2017-Broken Authentication	**SC-23 - Session Authenticity** AC-10 – Concurrent Session Control AC-11 – Session Lock AC-14 – Permitted Actions Without Identification or Authentication
A3:2017-Sensitive Data Exposure	**SI-11 – Error Handling** SC-4 - Information in Shared Resources SC-28 – Protection of Information at Rest
A4:2017-XML External Entities (XXE)	SI-10 – Information Input Validation
A5:2017-Broken Access Control	**AC-3 – Access Enforcement** SI-10 – Information Input Validation CM-7 Least Functionality
A6:2017-Security Misconfiguration	CM-6 – Configuration Settings
A7:2017-Cross-Site Scripting (XSS)	SI-10 – Information Input Validation
A8:2017-Insecure Deserialization	SC-23 - Session Authenticity
A9:2017-Using Components with Known Vulnerabilities	SI-2 – Flaw Remediation
A10:2017-Insufficient Logging&Monitoring	**AU-12 Audit Generation** AU-6 Audit Review, Analysis, and Reporting

Acronyms

Acronym	Definition
AC	Access Control
ACT	Adaptive Capabilities Testing
ATO	Authorization to Operate (also: Authority to Operate)
CCE	Common Configuration Enumeration
CCIC	CMS Cybersecurity Integration Center
CD	Continuous Delivery (also: Continuous Deployment)
CFACTS	CMS FISMA Controls Tracking System
CI	Continuous Integration
CIO	Chief Information Officer
CIS	Center for Internet Security
CISO	Chief Information Security Officer
CM	Configuration Management
CMS	Centers for Medicare & Medicaid Services
CRA	Cyber Risk Advisor
CVE	Common Vulnerabilities and Exposures
CWE	Common Weakness Enumeration
DevOps	Development (and) Operations (working together)
DevSecOps	Development, Security, Operations (working together)
DISA	Defense Information Systems Agency
DORA	DevOps Research and Assessment
FISMA	Federal Information Security Modernization Act (2014)
IA	Identification and Authentication
ISPG	Information Security and Privacy Group
ISSO	Information System Security Officer
JSON	JavaScript Object Notation
NIST SP	National Institutes of Standards and Technology Special Publication
OWASP	Open Web Application Security Project
RBAC	Role-based Access Control

Acronym	Definition
SANS	SysAdmin, Audit, Network, and Security
SIA	Security Impact Analysis
SSP	System Security Plan
STIG	Security Technical Implementation Guide