“Replace with Organization logo”

<Identify your organization>

DevSecOps Concept of Operations

(DSOP CONOPS)

Version v0.1

Last Update: 5/24/2023

**Distribution Statement:** Must be completed when filled in.

**DESTRUCTION NOTICE**: Must be completed when filled in.

# Version History

REVISION AND HISTORY PAGE Revisions and version changes to this document are recorded within the following table. New versions are published when changes to the document equate to 10 percent or greater of the document’s content, or if a change requires immediate implementation. This record is maintained throughout the life of the document.

This document will be reviewed at a minimum annually.

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Change Type** | **Modified By** |
| 24 May 2023 | V0.1 | Initial Version | ARLO-Solutions |

# **Table of Contents**

[Table of Contents 3](#_Toc69981552)

[How to Use This Document 6](#_Toc69981553)

[*I.* *Executive Overview* 7](#_Toc69981554)

[1. Introduction 7](#_Toc69981555)

[2. Purpose 7](#_Toc69981556)

[3. Background 8](#_Toc69981557)

[4. Project/Program Overview 8](#_Toc69981558)

[5. Assumptions and Constraints 8](#_Toc69981559)

[*II.* *The Infrastructure* 8](#_Toc69981560)

[6. Operating Environment 9](#_Toc69981561)

[6.1. Current System(s) 9](#_Toc69981562)

[6.2. Future System(s) 9](#_Toc69981563)

[6.3. Organizations and Personnel 9](#_Toc69981564)

[6.4. Stakeholders 10](#_Toc69981565)

[7. Security Operations 10](#_Toc69981566)

[7.1. Team Structure and Support 10](#_Toc69981567)

[7.2. Cyber Security Service Provider (CSSP) Services 10](#_Toc69981568)

[8. System Operational Overview 11](#_Toc69981569)

[8.1. Networking Infrastructure 11](#_Toc69981570)

[8.2. Information Transfer and Collaboration 11](#_Toc69981571)

[8.3. Hardware 12](#_Toc69981572)

[8.4. Software 12](#_Toc69981573)

[8.5. Tools 12](#_Toc69981574)

[8.6. Securing the System 13](#_Toc69981575)

[8.7. Maintenance 13](#_Toc69981576)

[9. Physical Security 14](#_Toc69981577)

[9.1. System/Facility Access 14](#_Toc69981578)

[9.2. Physical Environment 14](#_Toc69981579)

[9.3. Data Storage Media 14](#_Toc69981580)

[10. Data Storage 14](#_Toc69981581)

[11. Backup and Recovery 15](#_Toc69981582)

[*III.* *The CI/CD Pipeline* 15](#_Toc69981583)

[12. Prepare the Organization 16](#_Toc69981584)

[12.1. Identify Security Requirements for Software Development 17](#_Toc69981585)

[12.2. Roles and Responsibilities 17](#_Toc69981586)

[12.2.1. Define New Roles and Responsibilities 18](#_Toc69981587)

[12.2.2. Provide Role Specific Training 18](#_Toc69981588)

[12.3. Supporting Toolchain 18](#_Toc69981589)

[12.3.1. Specify Tools 19](#_Toc69981590)

[12.3.2. Security Practices and Tool Configuration 20](#_Toc69981591)

[12.4. Criteria for Software Security 20](#_Toc69981592)

[12.5. Define Criteria 20](#_Toc69981593)

[12.5.1. Implement Processes 20](#_Toc69981594)

[13. Protect the Software 21](#_Toc69981595)

[13.1. Protect Code from Unauthorized Access and Tampering 21](#_Toc69981596)

[13.1.1. Store Code Securely 21](#_Toc69981597)

[13.2. Software Release integrity 21](#_Toc69981598)

[13.2.1. Verification Information Availability 22](#_Toc69981599)

[13.3. Software Release Protection 22](#_Toc69981600)

[13.3.1. Securely Archive each Release 22](#_Toc69981601)

[14. Produce Well Secured Software 22](#_Toc69981602)

[14.1. Design Software to Meet Security Requirements 22](#_Toc69981603)

[14.1.1. Risk Modeling 22](#_Toc69981604)

[14.2. Verify Software Design Complies with Security Requirements 23](#_Toc69981605)

[14.2.1. Security Review of Software Design 23](#_Toc69981606)

[14.3. Evaluate Third Party Software 23](#_Toc69981607)

[14.3.1. Work with Third Parties 24](#_Toc69981608)

[14.3.2. Commercial and Open Source 24](#_Toc69981609)

[14.4. Reuse of Existing, Well Secured Software 24](#_Toc69981610)

[14.4.1. Acquisition of Secure Software Components 25](#_Toc69981611)

[14.4.2. In-House Creation of Well Secured Software 25](#_Toc69981612)

[14.4.3. Standardize Security Features 25](#_Toc69981613)

[14.5. Source Code Adheres to Secure Coding Practice 26](#_Toc69981614)

[14.5.1. Secure Coding Practices 26](#_Toc69981615)

[14.5.2. Analyze Human Readable Code 26](#_Toc69981616)

[14.6. Compilation and Build Process 26](#_Toc69981617)

[14.6.1. Compiler and Build Tools 26](#_Toc69981618)

[14.6.2. Build Tool Features Used 27](#_Toc69981619)

[14.7. Review and Analyze Human Readable Code 27](#_Toc69981620)

[14.7.1. People and/or Tools for Code Review/Analysis 27](#_Toc69981621)

[14.7.2. Process for Code Review/Analysis 28](#_Toc69981622)

[14.8. Test/Verify Executable Code 28](#_Toc69981623)

[14.8.1. Executable Code Testing 28](#_Toc69981624)

[14.8.2. Design and Perform Testing 29](#_Toc69981625)

[14.9. Secure Settings by Default 29](#_Toc69981626)

[14.9.1. Configuration for Each Setting 29](#_Toc69981627)

[14.9.2. Implement Default Settings 30](#_Toc69981628)

[15. Respond to Vulnerabilities 30](#_Toc69981629)

[15.1. Identify and Confirm Vulnerabilities 30](#_Toc69981630)

[15.1.1. Information from Consumers and Public Sources 30](#_Toc69981631)

[15.1.2. Review, Analyze and/or Test Code 30](#_Toc69981632)

[15.1.3. Team and Process 31](#_Toc69981633)

[15.2. Assess, Prioritize and Remediate Vulnerabilities 31](#_Toc69981634)

[15.2.1. Analyze Vulnerabilities 31](#_Toc69981635)

[15.2.2. Remediation Plan 31](#_Toc69981636)

[15.2.3. The SDLC Process 32](#_Toc69981637)

[*IV.* *Information System Continuous Monitoring (ISCM)* 32](#_Toc69981638)

[*V.* *Configuration Management (CM)* 33](#_Toc69981639)

[*VI.* *Incident Response (IR)* 33](#_Toc69981640)

*Disclaimer: This document is meant to be used as a template only and can be modified at the program’s discretion. It is not meant to be an all-inclusive or conclusive guide to how your DSOP environment should operate. The program is responsible for addressing the “How” details of the DSOP environment (i.e., system functionality, internal processes, procedures, and risk determinations). This CONOPS provides a foundation to address all areas within the DSOP program.*

# How to Use This Document

*A pre-requisite to successfully completing this CONOPS, is being familiar with the “DoD CIO DevSecOps Reference Design”, the NIST “Mitigating the Risk of Software Vulnerabilities by Adopting a Secure Software Development Framework (SSDF)”, and the “Adaptive Acquisition Framework”.*

*This document includes* black text *to indicate instructions to best create a DSOP Concept of Operations (CONOPS) document. Each set of instructions may be accompanied by supplemental information in blue italicized text, or Tips for Success, to include best practices for creating the most effective and detailed CONOPS for your organization.*

*To turn the template into your actual CONOPS, replace the* black text *after each heading with the information specifically for your organization. Use this template as a* ***guide*** *to document your DSOP operation.*

*The guide is divided into a number of Sections and Sub-Sections. The scope of the initiative will depict which Sections need to be completed.*

*I. Executive Overview (Sec. 1-5)*

*II. The Infrastructure: This includes* ***System, Platform, People and Processes*** *that operate, maintain, secure and encompass the overall environment. Complete if building a DSOP Platform, however if building a pipeline to be hosted on a 3rd party managed platform defer to the existing documentation and complete accordingly.*

*III. The Continuous Integration/Continuous Delivery (CI/CD) Pipeline: This section is to document the Tools, Process and People that execute and utilize the pipeline – The initial pipeline main purpose is to maintain the platform. If applicable, any additional pipelines that are not meant to maintain the platform must address their specific process, people and tools utilized along with the external system they are supporting and/or the target for their product. Those pipelines will inherit “The Infrastructure”.*

*IV. Information System Continuous Monitoring (ISCM) – This section is meant to address how ISCM is being implemented to monitor both the system and the pipeline. Describe the strategy and process that is relevant to the applicable component(s) of the infrastructure within your area of responsibility.*

*V. Configuration/Change Management (CM) - This section is meant to address how CM is being implemented on both the system and the pipeline. Describe the CM implementation relevant to the applicable component(s) of the infrastructure* *within your area of responsibility.*

*VI. Incident Response (IR) – Complete for the Infrastructure. The pipeline does not need to address IR, but rather within the pipeline, address responses to detected and reported vulnerabilities of the product.*

# Executive Overview

# **Introduction**

Provide a brief, high-level description of the system. Describe, from a high-level holistic view, what your organization’s mission is. Briefly describe what your goals are with the completion of the CONOPS. Provide a full identification of the information system to which this document applies, including, as applicable, categorization, identification number(s), title(s), abbreviations(s), version number(s), effectiveness, and release number(s) (e.g., eMASS, ITIPS, ITSC).

*Tips to success: The DSOP execution requires adherence to specific design principals, processes, and mechanisms for technical and non-technical activities, including engineering, design, development, testing, and operations and maintenance (O&M). Ensure that all associated roles and people are identified, and that these stakeholders are equipped with the proper skills, training, and experience, and are given access to the necessary training, policies, procedures, documentation, and other artifacts to accomplish their duties. In developing the Security CONOP for a DSOP platform and Continuous Integration / Continuous Delivery (CI/CD) pipeline one must be familiar with the DoD CIO DevSecOps Reference Design. Compliance with that design is the first gate when evaluating such a system.*

# **Purpose**

Briefly describe the purpose of the system, the environment, and overall goal(s). State whether this DevSecOps CONOPS is intended to lead to an Authority to Operate (ATO) (e.g., full system approval under Risk Management Framework.) or ATO with Conditions (e.g., for a limited time period) or Interim Approval to Test (IATT) for development and testing or continuous Authority to Operate (cATO). Provide the planned development and authorization schedules.

# **Background**

Provide the impetus and rationale for the project/program, and its justification (e.g., cost savings, time savings, other). If there is a history of motivating factors that drive the mission focus, please include this in the background of the CONOPS as it is helpful to understand the goals and purpose.

# Project/Program Overview

Provide a description of the system’s overall functionality. Give a brief description of what is to be implemented, when it is to be implemented, and where will it be implemented.

Describe the general nature of the system from a security perspective; summarize the history of system development, operation, and maintenance. Describe which portions of the system already exist, whether portions will be changed, and which are to be developed. Identify current and planned operating sites.

Describe the environment the system inhabits and the other systems with which it interacts.

Describe the system boundary.

# Assumptions and Constraints

Include any assumptions and constraints (e.g., assumptions about the development and authorization schedule, the envisioned Protection Levels (PL) and Levels of Concern (LoCs)). Include any assumptions and constraints regarding the security concepts, including connected systems. Indicate whether a co-utilization or joint processing request and approval is required.

*Tip for Success: Within this section document any dependencies such as inheritance, Cross Domain Solution (Approval), interconnection agreements/approvals.*

# The Infrastructure

*The purpose of this section is to outline the Concept of Operations by which the program builds, administers, and manages a DSOP platform environment. This includes program managers, security teams, operations teams, developer teams, software factory teams, and other relevant stakeholders.*

*Although various authorization boundaries may exist (e.g., hosting and/or servicing environment) emphasis on building, maintaining, and securing the infrastructure. It implements the ability to protect the infrastructure and provide a stable and secure platform for DSOP practices and CI/CD pipelines for development teams. (e.g., such as doing testing and validation in a separate standalone environment.)*

*Within this document, define how the platform was built, where it is built and how it is secured. Describe how the system will isolate the supporting pipeline from additional pipelines and teams. Additional pipelines that do not support the platform should not be able to deploy to the platform thus compromising the platform or other pipelines.*

# Operating Environment

## Current System(s)

Describe the current system (If no current system exists, mark as None or N/A). Where is the system hosted (e.g., DECC, Milcloud, AWS, Azure)? What are the service offerings being used (e.g., PaaS, IaaS, SaaS, physical hardware, virtual hardware etc. (note that in hybrid; one, some or all may be used))?

*Tips for Success: For a functional DSOP CI/CD Pipeline environment it must meet the Minimum Viable Product (MVP) as defined in the DoD CIO DevSecOps Reference Design. The MVP is about the software tools that must exist within the platform. What the organization considers their MVP may exceed the DoD CIO MVP.*

## Future System(s)

Describe what the future of the system will look like. Provide plans and schedules for additions to the current system.

*Tips for Success: Address in detail the timelines and resources involved in implementing infrastructure and tools beyond the MVP. The DoD CIO MVP is necessary to be considered a DSOP environment. The efficiency and abilities of the environment will be greatly improved with the addition of tools.*

## Organizations and Personnel

Describe the organizations and personnel that will use the system. Describe the proposed user group concept. Define the general users, privileged users, operators and acquirers of the system (clients).

*Tips for Success:*

*• Be sure to include all users and/or groups. Operations and Security teams along with user community morphs when implementing a DSOP environment. The expertise goes beyond maintaining servers and patching software (e.g., keeping the lights on). Infrastructure as Code (IaC) and Configuration as Code (CaC) introduces Developer and Software Factory teams and expand the training and knowledge requirements of the Operations and Security personnel.*

*• Address who the userbase will be (e.g., US only, Foreign Nationals, Industry Partners, etc.)*

## Stakeholders

List the certifier(s) and accreditor(s). Describe each stakeholder’s security concerns, particularly addressing Confidentiality, Integrity, and Availability (C, I, A), as necessary.

# Security Operations

## Team Structure and Support

Describe the structure, functions, roles, and responsibilities for the Security Operations (DevSecOps) Team (e.g., SOC, Security Testing, ISSO, ISSE, CRA et.). Use tables or diagrams as needed.

*Tips for Success: Address roles and responsibilities for each security member. Address the support frame they are active (e.g., 24/7, 0600 to 1800, on call.) Address what is being performed in house separate to what is being supplied by the Cyber Security Service Provider (detailed in next section)*

## Cyber Security Service Provider (CSSP) Services

Describe the services being supplied by the CSSP. If you are providing your own services, please explain what services are being conducted and please include what DoD support has been approved or needed. Include Service Level Agreement (SLA) as an attachment.

*Tips for Success: Per DoD Mandate 17-0019, DoDI 8530.01 you need a Cybersecurity Service Provider (CSSP), or your organization needs to be able to support similar services that meet the DoD requirements. CSSP are required for both on premises and cloud Information Systems. Establish Cyber Defense (CD) services through one of the 23 DoD certified Cyber Security Service Providers (CSSP) for Mission Cyber Defense (MCD).*

*CSSP services may include but are not limited to:*

*• Penetration Testing*

*• External Vulnerability Scans (EVS)*

*• Web Vulnerability Scanning (WVS)*

*• Malware Notification Protection (MNP)*

*• Support and Training (S&T)*

*• Network Security Monitoring (NSM)*

*• Attack Sensing & Warning (ASW)*

*• Warning Intelligence (WI)*

*• Incident Reporting (IR)*

*• Incident response Support (IRS)*

*• Volatile Data Analysis (VDA)*

*• Forensic Media Analysis (FMA)*

*• Reverse Engineering and Malware Analysis (RE/MA)*

*• Cyber Hunt / Intrusion Assessment (IA)*

*• Incident Response (IR)*

*• Sustainment & Configuration Management (SCM)*

*• HBSS or DoD approved equivalent Anti-Virus*

*• Port Whitelisting through DISA*

*• (Enterprise) Break and Inspect (E(BI))*

# System Operational Overview

Provide a summary description of the system’s operations. As needed use diagrams to describe the operations concept.

Provide a summary description of the Security Concept of Operation (CONOPS). Provide a brief description of the security rules the Security Support Structure shall enforce. Provide a brief summary of system start-up and shut-down procedures. If the system will operate in various configurations or modes provide a description of the overall security baseline and for each configuration. Also provide a summary of security procedures controlling the mode or configuration change.

## Networking Infrastructure

Identify and justify the proposed (C, I, A) and LoC. Identify and justify the proposed PL. Provide any factors used in determining the LoC and PL. Identify the periods the system will be operational (e.g., only during business hours, always (e.g., unattended processing)).

List other systems that will be affected and describe existing system connectivity.

*Tips for success: In a cloud-based environment much of the network architecture is provided by the Cloud Service Provider (CSP) but you have a responsibility to maintain the routing, connections and rules that govern the architecture. For a DSOP Environment the use of Infrastructure as code (IaC) and Configuration as Code (CaC) is required for a DSOP environment seeking a continuous Authority to Operate (c-ATO). The ability to stand up, expand and restore network, software and operating environments in milliseconds is paramount to success.*

## Information Transfer and Collaboration

Identify internal and external interfaces, Memorandum(s) of Understanding (MOU), and Facility Co-Use requirements. If a separate Security Architecture [Description] is available, this paragraph should refer to that document. Otherwise, provide a summary description of the system, operating system, key components, perimeter (physical and electrical) and user population. Provide a description of data flow.

If the system will operate in various configurations, modes or classification levels provide a description of each architecture configuration. Provide a summary of procedures controlling the configuration, mode, or classification change. Additionally, identify and address the Cross Domain Solutions (CDS) (e.g., Diode) implemented, with the processes for moving the information, along with direction of flow.

*Tips for Success: A number of documents and agreements may need to exist depending on organization(s) requirements that go beyond an MOU. Do not overlook Interconnection Service/Security Agreements (ISA) and the SLA to support the connection(s).*

## Hardware

Describe the system’s hardware and/or virtual assets. (e.g., physical servers, firewalls, Gateways, EC2 instances, virtual networking, virtual load balancers.)

*Tips for success: Hardware is used as a generic placeholder. The hardware is invisible to most system owners in hosting Data Centers (DC) and especially in the cloud. Hosted DCs generally provide you with a virtual environment to build in, seldom giving you access to the hypervisor. Your assets in this case are likely all virtual machines running Windows or Linux. For a cloud infrastructure your idea of hardware would be the EC2 instances and virtual load balancers that you have control of the configuration. Generally, look at anything that can have an IP address as hardware.*

## Software

Describe the system’s software (e.g., operating system, applications, management, etc.)

*Tips for success: When addressing software, you want to cover all operating systems (Ubuntu, RHEL, Windows etc.) operating within the environment. Cover all identifiable applications to include developer tools, testing tools, management tools, user applications and how these products are deployed (software running on a EC2 instance, software running in a container etc.) Indicate the purpose for each software and tool. (authentication, authorization, development, testing, monitoring etc.)*

## Tools

Describe the tools being utilized throughout your CI/CD pipeline.

*Tips for success:*

*• What are you using for the CI/CD and/or container orchestrator? (e.g., Kubernetes, GitLab, Jenkins)*

*• What tools have you chosen to utilize for Code Repository? (e.g., GIT)*

*• What tools have you chosen to accomplish the Static Code Analysis? (e.g., Surity, Fortify, SonarCube)*

*• What application specific tools are you using for Unit testing? (e.g., Java, NodeJS, Python, .NET)*

*• What tools are you utilizing for conducting your Build Image process? (e.g., Docker, GitLab)*

*• What tools are you utilizing for conducting your Dynamic Application Security Test? (e.g., Anchore, Twistlock, Surety)*

*• What tool are you using for Interactive Application Security Test (IAST)?*

*• What tools are you utilizing to complete your Run Test Environment? (e.g., Surety, Zapproxy, Cypress.io)*

*• What tools are used for application specific testing, fuzz testing, performance testing, and integration testing? (e.g., Cypress, OWASP ZAP)*

*• What tools are utilized to complete code acceptance? (e.g., Kubernetes, Argo, Kustomize.io)*

*• What tools will be used for code release? (e.g., HIL, GitLab, ArtiFactory)*

*• What tools will be used for managed services? (e.g., Docker, Kubernetes, Ansible, GitLab, RedHat OpenShift)*

## Securing the System

Describe how security is being applied to the system’s network, hardware and software.

*Tips for success:*

*• Address how images and configurations are secured. Address how IaC and CaC are being used and how those code repositories are secured. Address how IAC and CaC are being reviewed for vulnerabilities. Address application of STIGs and hardening of the operating systems, software, tools, network, and containers.*

*• Address what is being used for monitoring such as Host Based Intrusion Detection/Prevention (HIDS/HIPS), Network Intrusion Detection/Prevention (NIDS/NIPS), and Security Event and Incident Management (SEIM) solution.*

## Maintenance

Describe how the hardware, software, and interfaces will be maintained and sustained.

*Tips for success:*

*• In many environments the physical infrastructure is maintained by the hosting facility or CSP. You are responsible to maintain your images, patch operating systems, patch software and maintain the network layout. Much of this should be done by IaC and CaC. Explain how the environment is maintained, kept up to date and secured.*

*• Look at the organizations overall Operations & Maintenance (O&M) process and how it would apply.*

*• Consider, think and identify risk associated with your supply chain for acquiring software, hardware, support etc. Look at process for maintaining:*

*• Bill of Material (BOM)*

*• Existing supplier management identifies supplier source, End of Life (EOL) analysis,*

*and alternate part analysis. (Document “As-Is”)*

*• Existing criteria being used by primes and flowed down to subs, on purchasing of parts*

*is known.*

*• What is the supply chain mapping? Does one exist already?*

*• Available intel/ threat info that can be applied against the list of parts or suppliers*

*identified (or technologies).*

# Physical Security

***Note:*** *For cloud hosted environments this section can be summarized noting the Cloud Service Provider (CSP) and the FEDRAMP Certification and Impact Level.*

## System/Facility Access

Describe the users and their data access levels and who has facility access. Include personnel access controls, after hour’s access, and procedures for providing access to uncleared visitors (e.g., admitting, area sanitizing, escorting). Include a floor plan and an equipment layout and any protected distribution systems. Identify areas where other or lower classified systems are collocated with this system. Provide supporting diagrams as necessary.

## Physical Environment

Describe the site/facility, its location, and whether it is accredited. Include information about the measures used to protect the facility. Describe procedures for controlling physical access to the facility and to the system.

## Data Storage Media

Describe media labeling, transport, and release. How is media received and released from the facility? What is the labeling and transport requirements? How is it secured internally and in transport?

# Data Storage

Describe the organizations data storage process for the system:

What are the data types being stored? (e.g., Mission based, management and support, legislative and executive, etc.)

How is the data secured and maintained?

Who or what has access to the data storage?

What are the Retention periods for the data types?

*Tips for Success:*

*• Think broadly when detailing data storage. The cloud expands this idea tremendously. The existence of data lakes for high-speed big data analytics that see high volumes of queries, compared to data stores that are meant for archive and retrieval compared to those that are meant for storage and rarely accessed. Data is stored everywhere and anywhere; nothing can be overlooked. For example; identifying if the data will be ABAC/RBAC/MAC controlled. Especially in light of the identified user base that will have access, such as foreign nationals or foreign partners.*

*• For Information Data Types ref. NIST SP800-60 vol. 1 Guide for Mapping of Information and Information Systems to Security Categories.*

# Backup and Recovery

Describe the organizations backup and recovery process for the system.

Where are backups being stored?

How are backups secured and maintained?

What are the retention periods?

Explain your disaster recovery exercise types, how often performed and when last performed.

*Tips for Success:*

*• This section may add to data storage. Most backups are kept in low-cost storage that is not optimized for daily activity or transactions and you may have addressed that storage in Sec. “Data Storage”.*

*• Use this section to detail how your organization performs backup and recovery from the possibly various storage mediums. Address your requirements to perform backup and recovery exercises to verify your process works and your backups will restore you to operations*

# The CI/CD Pipeline

*The purpose of this Section is to outline the Concept of Operations by which the program administers the people and processes associated with execution, operation, security and the integrity of the product that comes out of the pipeline. This includes program managers, software developers, systems administrators, and other relevant stakeholders.  
  
This document focuses on implementing the practices, processes and people, and identifies the tools and techniques used to do so. Any organization’s developer using this pipeline is required to follow the people and process controls and guidance as defined in this section.  
  
Within this document, define how the fundamental, sound, and secure software development practices are integrated into the organization and CI/CD pipeline.*

*The practices are organized into four groups:*

*1. Prepare the Organization (PO): Ensure that the organization’s people, processes, and technology are prepared to perform secure software development at the organization level and, in some cases, for each individual project.*

*2. Protect the Software (PS): Protect all components of the software from tampering and unauthorized access.*

*3. Produce Well-Secured Software (PW): Produce well-secured software that has minimal security vulnerabilities in its releases.*

*4. Respond to Vulnerabilities (RV): Identify vulnerabilities in software releases and respond appropriately to address those vulnerabilities and prevent similar vulnerabilities from occurring in the future.*

*The Pipeline Section is organized to align in context with the sections of the NIST Secure Software Development Framework (SSDF). Additional helpful information can be found in NIST SP 800-137 Information System Continuous Monitoring (ISCM) for Federal Information Systems and Organizations DoD CIO DevSecOps Reference Design.*

# Prepare the Organization

Define how the organization is preparing the people, process, and technology to perform secure software development at the organization level and, in some cases, for each individual project.

*Tip for Success: Security requirements for software development must always be known. In considering the Software Development Life Cycle (SDLC) requirements from internal sources (e.g., the organization’s policies, business objectives, and risk management strategy) and external sources (e.g., applicable laws and regulations), minimize the duplication of effort by streamlining the processes, by collecting and sharing once.*

## Identify Security Requirements for Software Development

Identify all applicable security requirements for the organization, general software development and provide a brief description of how you are maintaining the requirements over time.

• Define policies that specify the security requirements for the organization’s software to meet, including secure coding practices for developers to follow.

• Define policies that specify software architecture requirements, such as making code modular to facilitate code reuse and easier updates as well as isolating security functionality from other functionality during code execution.

• Define policies for securing the development infrastructure, such as developer workstations and code repositories.

• Define the requirements for continuously monitoring the SDLC.

• Define the requirements for configuration and change management (CM) throughout the SDLC.

*Tips for Success:*

*• Certify the policies cover the entire software life cycle, including notifying users of the impending end of software support and the date of software end-of-life.*

*• Use a well-known set of security requirements as a structure or lexicon for defining the organization’s requirements. This set can be mapped to other third-party security requirements to which the organization is also subject.*

*• Review and update the requirements after each response to a vulnerability incident.*

*• Conduct a periodic (typically at least annual) review of all security requirements.*

*• Promptly review new external requirements and updates to existing external requirements.*

*• Educate affected individuals on the impending changes in requirements.*

## Roles and Responsibilities

Identify the individual(s) or group(s) with a primary responsibility and have a vested interest in the environment. Ensure that all stakeholders (internal and external) are integrated in the SDLC process and are prepared to perform their related tasks.

*Tips to Success: Take a top-down approach*

*• Increase awareness by upper management.*

*• Assist upper management in incorporating secure development support into their communications with personnel with SSDF, ISCM and CM related roles and responsibilities.*

*• Educate all personnel with SSDF, ISCM and CM related roles and responsibilities on upper management’s commitment to the organization development of a Continuous Integration / Continuous Delivery (CI/CD) pipeline.*

### Define New Roles and Responsibilities

Following the Secure Software Development Framework (SSDF), define any new roles and responsibilities. Periodically review the defined roles and responsibilities and update them as needed.

Define SSDF, ISCM and CM related roles and responsibilities for all members of the Software Development, Software Factory, Security, Operation and Management teams. These teams consist of cybersecurity staff, security champions, project managers and leads, senior management, software developers, software testers/quality assurance personnel, product owners, and others involved in the SDLC.

*Tips for Success:*

*• Integrate the security roles into all teams not just security team.*

*• Conduct at minimum annual review of all roles and responsibilities.*

*• Educate affected individuals on the impending changes in roles and responsibilities.*

### Provide Role Specific Training

Define role-specific training for all personnel with responsibilities that contribute to secure development. Include desired outcome of each role. Periodically review role-specific training and update as needed.

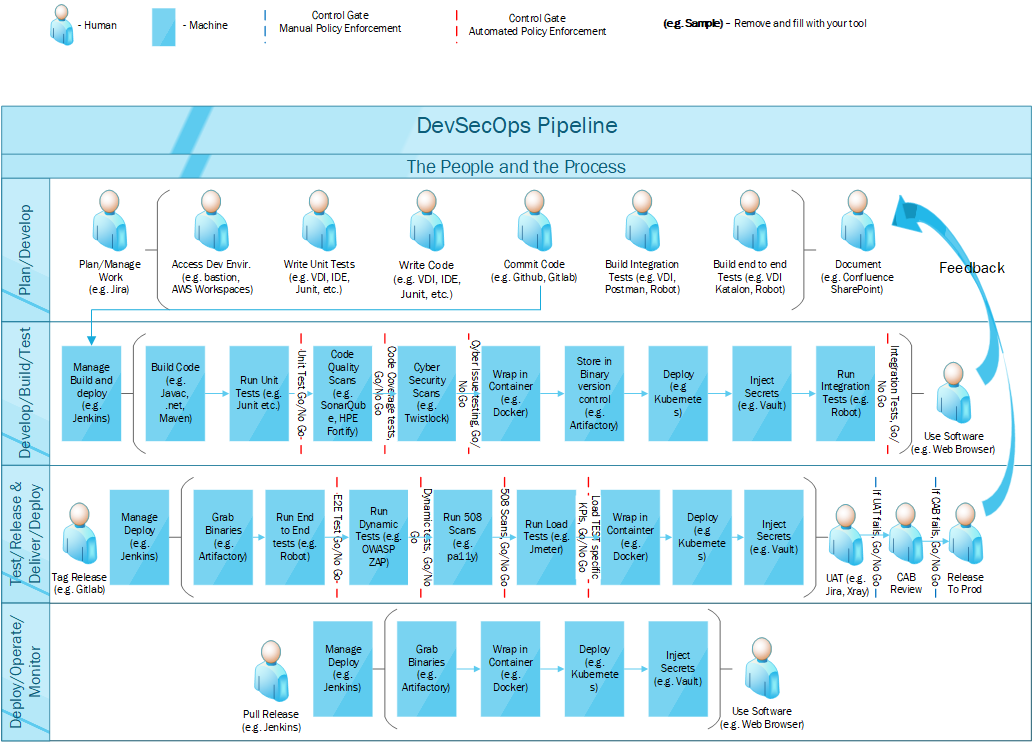
*• Create a training plan for each role.*

*• Acquire or create training for each role; acquired training may need customization for the organization.*

## Supporting Toolchain

Use of automation is required to reduce the human effort needed and improve accuracy, consistency, and comprehensiveness of security practices throughout the SDLC. Document the process, toolchains and tools being used at the different levels of the organization (e.g., organization-wide, or project-specific).

*Tips for Success: Example 0f a CI/CD toolchain*



### Specify Tools

Specify which tools and tool type(s) are to be included in each toolchain. In addition, define how the toolchain components are to be integrated with each other.

Define categories of toolchains and specify the mandatory tools or tool types to be used for each category. Categories should at minimum be CI/CD pipeline, ISCM process and the CM process. Categories can have subcategories. Identify security tools to integrate into the developer toolchain.

*Tips for success:*

*• The mandatory list should be the Minimal Viable Product (MVP) list. An additional objective list should be created to address tools that will be added as the system matures.*

*• Use automated technology for toolchain management and orchestration.*

*• Key when evaluating tools is the ability for the tools to interact with a common tool to minimize dashboards and enhance analytic ability.*

### Security Practices and Tool Configuration

Following sound security practices (ref., SSDF, Secure Cloud Computing Architecture (SCCA), etc.). Document the process to deploy and configure tools within the toolchain and how you maintain the individual tools and the toolchain, by following the SDLC.

*Tips for success:*

*• Evaluate, select, and acquire tools, and assess the security of each tool.*

*• Integrate tools with other tools and with existing software development processes and workflows.*

*• Update, upgrade, and replace existing tools.*

*• Monitor tools and tool logs for potential operational and security issues.*

*• Use the organization’s existing workflow or issue tracking systems to create an audit trail of the secure development-related actions that are performed.*

*• Determine how often the collected information should be audited, and implement*

## Criteria for Software Security

Define a software security baseline for validating the security checkpoint during development, to ensure that the SDLC output meets the organization’s defined software expectations.

## Define Criteria

Following the SSDF, define the criteria for software security checks and validation points (i.e., control gates) throughout the CI/CD and SDLC process.

• Define key performance indicators (KPIs) for software security.

• Define where in the CI/CD the control gates will be implemented.

*Tips for Success:*

*• Ensure that the criteria adequately indicate how effectively security risk is being managed.*

*• Add software security criteria to existing checks (e.g., the Definition of Done in agile SDLC methodologies).*

*• Review the artifacts generated as part of the software development workflow system to determine if they meet the criteria purposes.*

*• Record security check approvals, rejections, and requests for exception as part of the workflow and tracking system.*

### Implement Processes

Define the processes and mechanisms being used to gather the necessary information in support of the criteria.

*Tips for Success:*

*• Use the toolchain to automatically gather information that informs security decision-making.*

*• Deploy additional tools if needed to support the generation and collection of information supporting the criteria.*

*• Automate decision-making processes utilizing the criteria.*

# Protect the Software

## Protect Code from Unauthorized Access and Tampering

Define how you mitigate any unauthorized changes to the code (both inadvertent and intentional), which could circumvent or negate the intended security characteristics of the software. Having mitigations in place, helps deter and or prevent unauthorized changes and makes it more difficult for attackers to exploit vulnerabilities.

### Store Code Securely

Following the principle of least privilege, store all forms of code (i.e., source code, executable code), so that only authorized personnel have the necessary forms of access. Define how your organization meets these criteria.

*Tips for Success:*

*• Store all source code in a code repository and restrict access to it based on the nature of the code. For example, some code may be intended for public access, in which case its integrity and availability should be protected; other code may also need its confidentiality protected.*

*• Use version control features of the repository to track all changes made to the code with accountability to the individual developer account.*

*• Review and approve all changes made to the code.*

*• Use code signing to help protect the integrity and provenance of executables.*

*• Use cryptography (e.g., cryptographic hashes) to help protect the integrity of files.*

*• Create and maintain a software bill of materials (SBOM) for each software package created.*

*• Implement monitoring and alerting on repositories to track access and change.*

*• Train your team to understand more than just unauthorized access, but how to identify if the code has been changed.*

## Software Release integrity

Define the CI/CD software scanning checks (i.e., HP Fortify, SonarQube, Twistlock) for software integrity validation. Safeguard that the software that has been developed is legitimate and has not been tampered with.

### Verification Information Availability

How do you disseminate software output validation information to assure it is made available to consumers?

*Tips for Success:*

*• Post cryptographic hashes for release files on a well-secured website.*

*• Use an established certificate authority for code signing so consumers can confirm the validity of signatures.*

*• Periodically review the code signing processes, including certificate renewal and protection.*

## Software Release Protection

Define how you are identifying, analyzing, and mitigating vulnerabilities discovered in the software after release.

### Securely Archive each Release

Where and how are you archiving and what protection measures are in place to protect each release? Describe how you are securely archiving a copy of each release and all its components (e.g., code, package files, third-party libraries, documentation), and release integrity verification information.

*Tip for Success:*

*• Store all release files in a repository, and restrict access to them.*

# Produce Well Secured Software

## Design Software to Meet Security Requirements

Identify and describe how you evaluated the applicable security requirements for the software’s design; determine what security risks the software is likely to face during production operation and how those risks should be mitigated by the software’s design; and justify any cases where risk-based decisions conclude that security requirements should be relaxed or waived. Addressing security requirements and risks during software design (secure by design) helps to make software development more efficient.

### Risk Modeling

Describe the different forms of risk modeling the organization is using (i.e., threat, attack, or attack surface mapping), to assess the security risk for the software.

*Tips for Success:*

*• Train the development team (the security champions in particular) or collaborate with a threat modeling expert to create threat models and attack models and to analyze how to use a risk-based approach to address the risks and implement mitigations.*

*• Perform more rigorous assessments for high-risk areas, such as protecting sensitive data and safeguarding identification, authentication, and access control, including credential management.*

*• Review vulnerability reports and statistics for previous software.*

## Verify Software Design Complies with Security Requirements

Describe how you plan to ensure that the software will meet the security requirements and address the identified risk information. To do so, you may consider reviewing the software design to confirm it addresses all the security requirements. Reviewing the risk models created during software design to determine if they appear to adequately identify the risks. Review the software design to confirm that it addresses the risks identified by the risk models. Have the software’s designer correct failures to meet the requirements. Change the design and/or the risk response strategy if the security requirements cannot be met.

### Security Review of Software Design

Explain how a qualified person, who was not involved with the software design, will review and confirm that the design meets all the security requirements and addresses identified risks.

*Tips for Success:*

*• Recommend independent review of the software design to confirm that it addresses all of the security requirements.*

*• Review the risk models created during software design to determine if they appear to adequately identify the risks.*

*• Review the software design to confirm that it satisfactorily addresses the risks identified by the risk models.*

*• Have the software’s designer correct failures to meet the requirements.*

*• Change the design and/or the risk response strategy if the security requirements cannot be met.*

## Evaluate Third Party Software

Define a core set of security requirements, and include them in the acquisition documents, software contracts, and other agreements with third parties. Identify the security-related criteria for selecting commercial and open-source software. Explain how your organization will require the providers of commercial software modules and services to provide evidence that their software complies with the organization’s security requirements. By identifying these areas, it will reduce the risk associated with using acquired software modules and services, which are potential sources of additional vulnerabilities.

*Tips for Success:*

*• Knowledge of the Adaptive Acquisition Framework would be helpful in completing these section(s).*

### Work with Third Parties

How will your organization communicate requirements to third parties who may provide software modules and services to the organization for reuse.

*Tips for Success:*

*• Require the providers of commercial software modules and services to provide evidence that their software complies with the organization’s security requirements.*

*• Establish and follow procedures to address risk when there are security requirements that third-party software modules and services do not meet.*

### Commercial and Open Source

How is the organization planning to evaluate the commercial, open source, and all other third-party software modules and services to ensure that they comply with the requirements (i.e., Certificate to Field (CtF)).

*Tips to Success:*

*• Determine if there are publicly known vulnerabilities in the software modules and services that the vendor has not yet fixed.*

*• Validate each software module or service is still actively maintained, which should include new vulnerabilities found in the software being remediated.*

*• Determine a plan of action for each third-party software module or service that is no longer being maintained or available in the future.*

*• Use the results of commercial services for vetting the software modules and services.*

*\*\* The process here should tie to Section 14.7 Review and Analyze Human Readable Code and Sec. 14.8 Test/Verify Executable Code.*

## Reuse of Existing, Well Secured Software

Describe how the organization will review and evaluate third-party software components in the context of their expected use. If a component is to be used in a substantially different way in the future, perform the review and evaluation again with that new context in mind. Identify and establish an organization-wide software repository to host sanctioned and vetted open-source components. The organization must maintain a list of organization-approved commercial software components and component versions. Define the designated components that must be included for software to be developed. Where feasible reuse software instead of duplicating.

### Acquisition of Secure Software Components

Identify how the organization will acquire well-secured components (e.g., software libraries, modules, middleware, frameworks) from third parties for use by the organization’s software. Define a core set of security requirements and include them in the acquisition documents, software contracts, and other agreements with third parties.

*Tips for Success:*

*• Review and evaluate third-party software components in the context of their expected use. If a component is to be used in a substantially different way in the future, perform the review and evaluation again with that new context in mind.*

*• Establish an organization-wide software repository to host sanctioned and vetted open-source components.*

*• Maintain a list of organization-approved commercial software components and component versions.*

*• Designate which components must be included by software to be developed*

### In-House Creation of Well Secured Software

Explain how the organization will create well-secured software components in-house following SDLC processes to meet common internal software development needs that cannot be met by third-party software.

*Tips for Success:*

*• Follow the organization-established security practices for secure software development.*

*• Maintain an organization-wide software repository for these components.*

*• Designate which components must be included by software to be developed.*

### Standardize Security Features

Explain how the organization plans to standardize the security features and services (i.e., integrating with log management, identity management, access control, and vulnerability management systems) instead of creating proprietary implementations of security features and services.

*Tips for Success:*

*• Maintain an organization-wide software repository of modules for supporting standardized security features and services.*

*• Designate which security features and services must be supported by software to be developed.*

## Source Code Adheres to Secure Coding Practice

Identify how the organization plans to validate all inputs, and properly encode all output (i.e., avoid using unsafe functions and calls, error handling, and provide logging and tracing capabilities). By doing so the organization will decrease the number of security vulnerabilities in the software and reduce costs by eliminating vulnerabilities during source code creation.

### Secure Coding Practices

Explain how the organization will follow all secure coding practices that are appropriate to the development languages and environment.

*Tips for Success:*

*• Validate all inputs and validate and properly encode all output.*

*• Avoid using unsafe functions and calls.*

*• Handle errors gracefully.*

*• Provide logging and tracing capabilities.*

*• Use development environments with features that encourage or require the use of secure coding practices.*

*• Follow procedures for manually ensuring compliance with secure coding practices.*

*• Check for other vulnerabilities that are common to the development languages and environment.*

### Analyze Human Readable Code

Define how the organization shall analyze human readable code (i.e., Have the developer review their own human-readable code, analyze their own human-readable code, and/or test their own executable code to complement (not replace) code review, analysis, and/or testing performed by others).

## Compilation and Build Process

Explain how the organization plans to use up-to-date versions of compiler and build tools. Identify how the organization will validate the authenticity and integrity of compiler and build tools. By identifying the organization will decrease the number of security vulnerabilities in the software and reduce costs by eliminating vulnerabilities before testing occurs.

### Compiler and Build Tools

How does the organization plan to use compiler and build tools that offer features to improve executable security?

*Tips for Success:*

*• Use up-to-date versions of compiler and build tools.*

*• Validate the authenticity and integrity of compiler and build tools.*

*\*\* The process here should tie to Section 14.7 Review and Analyze Human Readable Code and Sec. 14.8 Test/Verify Executable Code.*

### Build Tool Features Used

Define which compiler and build tool features should be used and how each should be configured, then implement the approved configuration for compilation and build tools, processes, etc.

*Tips for Success:*

*• Enable compiler features that produce warnings for poorly secured code during the compilation process.*

*• Implement the “clean build” concept, where all compiler warnings are treated as errors and eliminated.*

*• Enable compiler features that randomize characteristics, such as memory location usage, that would otherwise be easily predictable and thus exploitable.*

*• Conduct testing to ensure that the features are working as expected and not inadvertently causing any operational issues or other problems.*

*• Verify that the approved configuration is enabled for compilation and build tools, processes, etc.*

*• Document information about the compilation and build tool configuration in a knowledge base that developers can access and search.*

## Review and Analyze Human Readable Code

Identify vulnerabilities by reviewing and/or analyzing human readable code so they can be corrected before the software is released to prevent exploitation. Using automated methods lowers the effort and resources needed to detect vulnerabilities. Human-readable code includes source code and any other form of code an organization deems as human readable.

### People and/or Tools for Code Review/Analysis

Explain how you determine whether code review (i.e., a person directly looks at the code to find issues) and/or code analysis (i.e., tools are used to find issues in code, either in a fully automated way or in conjunction with a person) should be used.

*Tips for Success:*

*• Follow the organization’s policies or guidelines for when code review should be performed and how it should be conducted. This includes third-party code and reusable code modules written in-house.*

*• Follow the organization’s policies or guidelines for when code analysis should be performed and how it should be conducted.*

### Process for Code Review/Analysis

Define how the organization will perform the code review and/or code analysis based on the organization’s secure coding standards, and document and triage all discovered issues and recommended remediations in the development team’s workflow or issue tracking system.

*Tips for Success:*

*• Perform peer review of code, and review any existing code review, analysis, or testing results as part of the peer review.*

*• Use peer reviews to check code for backdoors and other malicious content.*

*• Use peer reviewing tools that facilitate the peer review process and document all discussions and other feedback.*

*• Use a static analysis tool to automatically check code for vulnerabilities and for compliance with the organization’s secure coding standards, with a human reviewing issues reported by the tool and remediating them as necessary.*

*• Use review checklists to verify that the code complies with the requirements.*

*• Use automated tools to identify and remediate documented and verified unsafe software practices on a continuous basis as human-readable code is checked into the code repository.*

*• Identify and document the root cause of each discovered issue.*

*• Document lessons learned from code review and analysis in a knowledge base that developers can access and search.*

*• Identify process to obtain metrics (i.e., automated test coverage, bug(s) found/remediated, change failure rate, frequency of code rollback).*

## Test/Verify Executable Code

Explain how the organization will test executable code to identify vulnerabilities and verify compliance with security requirements, identify vulnerabilities so they can be corrected before the software is released to prevent exploitation. Using automated methods lowers the effort and resources needed to detect vulnerabilities. Executable code includes binaries, directly executed bytecode, directly executed source code, and any other form of code an organization deems as executable.

### Executable Code Testing

Determine if executable code testing should be performed and, if so, which types should be used.

*Tips for Success:*

*• Follow the organization’s policies or guidelines for when code testing should be performed and how it should be conducted. This includes third-party executable code and reusable executable code modules written in-house.*

### Design and Perform Testing

Define how the organization will design the tests, perform the testing, and document the results.

*Tips for Success:*

*• Perform robust functional testing of security features.*

*• Integrate dynamic vulnerability testing into the project’s automated test suite.*

*• Incorporate tests for previously reported vulnerabilities into the project’s automated test suite to ensure that errors are not reintroduced.*

*• Use automated fuzz testing tools to find issues with input handling.*

*• If resources are available, use penetration testing to simulate how an attacker might attempt to compromise the software in high-risk scenarios.*

*• Identify and document the root cause of each discovered issue.*

*• Document lessons learned from code testing in a knowledge base that developers can access and search.*

## Secure Settings by Default

Explain how the organization will configure the software to have secure settings by default. Improve the security of the software at the time of installation to reduce the likelihood of the software being deployed with weak security settings that would put it at greater risk of compromise.

### Configuration for Each Setting

Describe how the organization configures each setting that influences security so that the default settings are secure and do not weaken the security functions provided by the platform, network infrastructure, or services.

*Tip for Success:*

*• Conduct testing to ensure that the settings, including the default settings, are working as expected and are not inadvertently causing any security weaknesses, operational issues, or other problems.*

### Implement Default Settings

Explain how the organization will implement the default settings (or groups of default settings, if applicable), and document each setting for software administrators.

*Tips for Success:*

*• Verify that the approved configuration is in place for the software.*

*• Document each setting’s purpose, options, default value, security relevance, potential operational impact, and relationships with other settings.*

*• Document how each setting can be implemented by software administrators.*

# Respond to Vulnerabilities

## Identify and Confirm Vulnerabilities

Define the process to identify and confirm vulnerabilities associated with ongoing, continuous monitoring and detection. Ensure that vulnerabilities are identified and remediated, reducing the window of opportunity for attackers.

### Information from Consumers and Public Sources

What is the organizations procedures for obtaining information from consumers and public sources on potential vulnerabilities in the software and any third-party components that the software uses and investigate all credible reports?

*Tips for Success:*

*• Establish a vulnerability response program and make it easy for security researchers to learn about your program and report possible vulnerabilities.*

*• Define what threat intel sources will be used. (i.e., vulnerability databases, reports etc.)*

*• Monitor vulnerability databases, security mailing lists, and other sources of vulnerability reports through manual or automated means.*

*• Use threat intelligence sources to better understand how vulnerabilities in general are being exploited.*

### Review, Analyze and/or Test Code

Define how the organization will review code to identify or confirm the presence of previously undetected vulnerabilities.

*Tip for Success:*

*• Configure the toolchain to perform automated code analysis and testing on a regular basis.*

*\*\* The process here should tie to Section 14.7 Review and Analyze Human Readable Code and Sec. 14.8 Test/Verify Executable Code.*

### Team and Process

Define each role and responsibilities to handle the responses to vulnerability and incident handling.

*Tips for Success:*

*• Have a policy that addresses vulnerability disclosure and remediation, and implement the processes needed to support that policy.*

*• Have a security response playbook to handle a generic reported vulnerability, a report of zero-days, a vulnerability being exploited in the wild, and a major ongoing incident involving multiple parties.*

## Assess, Prioritize and Remediate Vulnerabilities

Define how the organization prioritizes identified vulnerabilities, performs the assessment on those vulnerabilities, and how remediation is performed to correct the vulnerabilities.

### Analyze Vulnerabilities

Define how the organization performs analysis on each vulnerability in order to gather sufficient information to plan its remediation.

*Tips for Success:*

*• Use issue tracking software (existing software, if available) to document each vulnerability.*

*• Estimate how much effort would be required to remediate the vulnerability, the potential impact of vulnerability exploitation, the resources needed to weaponize the vulnerability, if that has not already been done and any other relevant factors needed to plan the remediation of the vulnerability.*

*• Document the root cause of each discovered issue, lessons learned from root cause analysis in a knowledge base that developers can access and search, and lessons learned from root cause analysis in a knowledge base that developers can access and search.*

*• Add mechanisms to the toolchain to automatically detect future instances of the root cause.*

*• Review the software for other instances of the reported problem and proactively fix them rather than waiting for external reports. The process here ties to Section 8.5 Source Code Adheres to Secure Coding Practice and 14.7 Review and Analyze Human Readable Code.*

### Remediation Plan

Define the remediation plan for each vulnerability and how the plan is implemented to correct each vulnerability.

*Tips for Success:*

*• For each vulnerability, make a risk-based decision as to whether it will be remediated or if the risk will be addressed through other means (e.g., risk acceptance, risk transference), and determine how its remediation should be prioritized.*

*• If a permanent mitigation for a vulnerability is not yet available, determine how the vulnerability can be temporarily mitigated until the permanent solution is available, and add that temporary remediation to the plan.*

### The SDLC Process

Review and document the SDLC process and update it as appropriate to prevent (or reduce the likelihood of) the root cause recurring in updates to the software.

*Tips for Success*

*• Document lessons learned from root cause analysis in a knowledge base that developers can access and search.*

*• Plan and implement changes to your secure software development practices.*

# Information System Continuous Monitoring (ISCM)

ISCM has become a requirement across Department of Defense (DoD) for all systems. This section should not serve as the ISCM Strategy but rather provide how that strategy is implemented for monitoring the system/platform/pipeline this CONOPS is designed for. Explain how the organization conducts continuous monitoring and the tools being utilized to meet the ISCM requirements.

When establishing a DSOP environment with a CI/CD Pipeline and to achieve a cATO for the pipeline a well-defined ISCM is paramount to success.

*Tips for Success:*

*• Sample activities supporting monitoring include logging; log analysis & auditing; performance monitoring; security monitoring (including containers and VMs); asset inventory; configuration monitoring; database monitoring & security auditing.*

*• Metrics and thresholds are automatically gathered, calculated, and made visible. Example metrics include number of security defects identified pre-production; percent of code coverage from security testing; number of failed builds due to security checks; mean time to detection; mean time to resolution.*

*• What is your level and hours of support for ISCM, what is being handled by your organization vs. being handled by the CSSP? (Ref. sec 7.2 and cover here if not covered there)*

*• How are you utilizing your Host Based Intrusion Detection/Prevention (HIDS/HIPS), Network Intrusion Detection/Prevention (NIDS/NIPS), and Security Event and Incident Management (SEIM) solution.*

*• Are the various feeds from your logging, auditing HIDS/HIPS, NIDS/NIPS and SEIM tools going to singular points with useable dashboards that enable alerting and forensics?*

*• The ISCM process should integrate throughout this document. It will allow for a DSOP Pipeline to be considered under cATO.*

# Configuration Management (CM)

Define how the organization conducts configuration management (CM), which includes change management. Continuously controlling the CM process is vitally important to an Authorization and a critical must for a Continuous Authorization. Within each section/sub-section identify the components of the system, platform and pipeline that will be identified under CM and the people and the processes that would address and manage CM.

Utilize this section to address the high-level process and reference the additional documents that would support the CM.

*Tips for Success:*

*• The CM process should integrate throughout this document. It will allow for a DSOP Pipeline to be considered under cATO.*

*• Establish a chartered Change Control Board (CCB) and/or Change Approval Board (CAB) if neither exists. A CCB must be chartered and usually exist at the organization level. A CAB does not require charter and is responsible at the system level. The CAB is best to address internal changes and understand the system. The CCB is best to judge organizational changes.*

*• A CM Plan needs to identify assets, processes, procedures, policies, practices etc. that will fall under the CM process. An organization is at liberty to define what will be part of the CM process. Caution should be placed on not adding unnecessary items in a cover all approach or leaving out critical assets in a minimized approach.*

*• Guide for Security-Focused Configuration Management of Information Systems is found in NIST SP800-128.*

# Incident Response (IR)

Describe the organization’s incident response plan and process for the system.

*Tips for Success: Incident response can be a multi facet undertaking. It could be performed by the organization, by the CSSP or combination of both. Address all aspects by entities involved. The IR plan could be a separate organizational document that affects the system in which case briefly cover its scope here and provide the IR plan as an attachment.*