PROJECT DESCRIPTION

SOFTWARE SUPPLY CHAIN AND DEVOPS SECURITY PRACTICES

Implementing a Risk-Based Approach to DevSecOps

Murugiah Souppaya
Michael Ogata
Paul Watrobski

National Institute of Standards and Technology

Karen Scarfone

Scarfone Cybersecurity

November 2022

devsecops-nist@nist.gov

This revision incorporates comments from the public.

NIST | NATIONAL CYBERSECURITY CENTER OF EXCELLENCE

This document describes a problem that is relevant to many industry sectors. NCCoE cybersecurity experts will address this challenge through collaboration with a Community of Interest, including vendors of cybersecurity solutions. The resulting reference design will detail an approach that can be incorporated across multiple sectors.

ABSTRACT

DevOps brings together software development and operations to shorten development cycles, allow organizations to be agile, and maintain the pace of innovation while taking advantage of cloud-native technology and practices. Industry and government have fully embraced and are rapidly implementing these practices to develop and deploy software in operational environments, often without a full understanding and consideration of security. Also, most software today relies on one or more third-party components, yet organizations often have little or no visibility into and understanding of how these components are developed, integrated, deployed, and maintained, as well as the practices used to ensure the components' security. To help improve the security of DevOps practices, the NCCoE is planning a DevSecOps project that will focus initially on developing and documenting an applied risk-based approach and recommendations for secure DevOps and software supply chain practices consistent with the Secure Software Development Framework (SSDF), Cybersecurity Supply Chain Risk Management (C-SCRM), and other NIST, government, and industry guidance. This project will apply these DevSecOps practices in proof-of-concept use case scenarios that will each be specific to a technology, programming language, and industry sector. Both closed source (proprietary) and open source technology will be used to demonstrate the use cases. This project will result in a freely available NIST Cybersecurity Practice Guide.

KEYWORDS

cloud-native technology; cybersecurity supply chain risk management; DevOps; DevSecOps; secure software development; Secure Software Development Framework (SSDF); supply chain security

DISCLAIMER

Table of Contents

1 Executive Summary 3

Purpose 3

Scope 4

Assumptions/Challenges 5

Background 5

2 Scenarios 6

Scenario 1: Free and Open Source Software (FOSS) Development 6

Scenario 2: Closed Source Software Development 6

3 High-Level Architecture 6

Component List 6

Desired Security Capabilities 7

4 Relevant Standards and Guidance 7

5 Security Control Map 9

Appendix A References 17

Appendix B Acronyms and Abbreviations 18

1 Executive Summary

Purpose

DevOps brings together software development and operations to shorten development cycles, allow organizations to be agile, and maintain the pace of innovation while taking advantage of cloud-native technology and practices. Industry and government have fully embraced and are rapidly implementing these practices to develop and deploy software in operational environments, often without a full understanding and consideration of security.

DevSecOps helps ensure that security is addressed as part of all DevOps practices by integrating security practices and automatically generating security and compliance artifacts throughout the processes and environments, including software development, builds, packaging, distribution, and deployment. This is important for several reasons, including:

- reducing vulnerabilities, malicious code, and other security issues in released software without slowing down code production and releases;
- mitigating the potential impact of vulnerability exploitation throughout the software lifecycle, including when the software is being developed, built, packaged, distributed, deployed, and executed on dynamic hosting platforms;
- addressing the root causes of vulnerabilities to prevent recurrences, such as strengthening test tools and methodologies in the toolchain, and improving practices for developing code and operating hosting platforms; and
- reducing friction between the members of the development, operation, and security teams in order to maintain the speed and agility needed to support the organization's mission while taking advantage of modern and innovative technology.

There is increasing recognition that DevSecOps should also encompass software supply chain security. Most software today relies on one or more third-party components, yet organizations often have little or no visibility into and understanding of how these software components are developed, integrated, deployed, and maintained, as well as the practices used to ensure the components' security. DevSecOps practices can help identify, assess, and mitigate cybersecurity risk for the software supply chain [1].

This document defines a National Cybersecurity Center of Excellence (NCCoE) project. This project focuses on developing and documenting an applied risk-based approach and recommendations for DevSecOps practices. For the purposes of this project, the term "DevSecOps" refers to integrating security practices developed by the security team into existing pipelines (e.g., continuous integration/continuous delivery [CI/CD]) and existing toolchains used by developers and managed by operations teams. NIST's proposed approach for this project is similar to those used for the NIST Secure Software Development Framework (SSDF) [2] and the NIST Cybersecurity Framework [3]. This project is intended to help enable organizations to maintain the velocity and volume of software delivery in a cloud-native way and take advantage of automated tools. The project will also determine how the practices and tasks from the NIST SSDF can be implemented as part of a DevSecOps approach.

The project's objective is to produce practical and actionable guidelines that meaningfully integrate security practices into development methodologies. Industry, government, and other organizations could then apply the guidelines when choosing and implementing DevSecOps practices in order to improve the security of the software they develop and operate. That, in turn, would improve the security of the organizations using that software, and so on throughout the software supply chain. Additionally, the project intends to demonstrate how an organization can generate artifacts as a byproduct of its DevSecOps practices to support and inform the organization's self-attestation and declaration to conformance to applicable NIST and industry-recommended practices for secure software development and cybersecurity supply chain risk management.

The project will also strive to demonstrate the use of current and emerging secure development frameworks, practices, and tools to address cybersecurity challenges. Lessons learned during the project will be shared with the security and software development communities to inform improvements to secure development frameworks, practices, and tools. Lessons learned will also be shared with standards developing organizations to inform their DevSecOps-related work.

We received community feedback on the initial draft of this project description through both public comments and the NCCoE DevSecOps Workshop. We revised the project description to address the feedback.

This project will result in a publicly available NIST Cybersecurity Practice Guide, a detailed implementation guide of the practical steps needed to implement a cybersecurity reference design that addresses this challenge.

Scope

This project will apply DevSecOps practices in multiple proof-of-concept use case scenarios that each involve different technologies, programming languages, industry sectors, etc. The NCCoE project will use closed source and open source technology to demonstrate the use cases. The intention is to demonstrate DevSecOps practices, especially using automation, that would apply to organizations of all sizes and from all sectors, and to development for information technology (IT), operational technology (OT), Internet of Things (IoT), and other technology types. This project will not focus on the development of any particular technology type.

As part of this project, NIST will bring together and normalize content on DevSecOps practices from existing guidance and practices publications. This content, to be published as part of the project's NIST Cybersecurity Practice Guide, will be drafted and revised in parallel with the use case implementations. It will provide definitions of fundamental DevSecOps concepts so that developers, security professionals, and operations personnel can all have the same shared understanding of them. Also, it will document key elements that organizations would need to build successful DevSecOps practices, from changing the organization's culture to automating security practices into existing development pipelines and toolchains to support the concept of continuous authorization to operate (ATO). The guide will also provide all organizations with a way to document their current DevSecOps practices and define their future target practices as part of their continuous improvement processes. The recommendations and practices in the guide will be crafted to provide organizations choosing to adopt them with flexibility and customizability in their implementation.

Selected NIST guidance most closely related to DevOps and supply chain security, such as NIST Special Publication (SP) 800-218 [2], SP 800-190 [4], and SP 800-161 [1], will be leveraged for the use case implementations and may be updated during the course of the project based on lessons learned from the implementations. There are many existing security guidance and practices publications from NIST and others, but they have not yet been put into the context of DevOps or DevSecOps. Industry, standards developing organizations, government agencies, and others are already performing DevSecOps. Their efforts would be leveraged to provide a community-developed set of recommended practices. Updating affected NIST publications so they reflect DevSecOps principles would also help organizations to make better use of their recommendations.

Assumptions/Challenges

Readers are assumed to understand basic DevOps and secure software development concepts.

Background

A *software development life cycle (SDLC)*[1] is a formal or informal methodology for designing, creating, and maintaining software (including code built into hardware). There are many models for SDLCs, including waterfall, spiral, agile, and – in particular – agile combined with software development and IT operations (DevOps) practices. Few SDLC models explicitly address software security in detail, so secure software development practices usually need to be added to and integrated into each SDLC model. Regardless of which SDLC model is used, secure software development practices should be integrated throughout it for three reasons: to reduce the number of vulnerabilities in released software, to reduce the potential impact of the exploitation of undetected or unaddressed vulnerabilities by both external attackers and insider threats, and to address the root causes of vulnerabilities to prevent recurrences. Vulnerabilities include not just bugs caused by coding flaws, but also weaknesses caused by security configuration settings, incorrect trust assumptions, secrets or credentials stored within code, and outdated or incorrect risk analysis [5].

Most aspects of security can be addressed at multiple places within an SDLC, typically with some differences in cost, effectiveness, and ease of integration. However, in general, the earlier in the SDLC that security is addressed, the less effort and cost is ultimately required to achieve the same level of security. This principle, known as *shifting left*, is critically important regardless of the SDLC model. Shifting left minimizes any technical debt that would require remediating early security flaws late in development or after the software is in production, at a higher cost than if they'd been remediated earlier. Shifting left can also result in software with stronger security.

With today's software, the responsibility for implementing security practices is often distributed among multiple organizations based on the delivery mechanism (e.g., infrastructure as a service, software as a service, platform as a service, container as a service, serverless). In these situations, it likely follows a shared responsibility model involving the platform/service providers and the tenant organization that is consuming those platforms/services. The parties will need to agree on what security practices need to be performed based on the organization's defined policy, regulations, and mandates, which party is responsible for each practice, and how each party will attest to their conformance with the agreement.

Another aspect of today's software is that it often uses one or more software components developed by other organizations, groups, or individuals. Some of those components may also use components from other organizations, and so on. Managing cybersecurity risk from third-party software components, as part of cybersecurity supply chain risk management (C-SCRM), involves identifying, assessing, selecting, and implementing processes and mitigating controls. This risk management can largely be integrated into DevSecOps through its automation

[1] Note that SDLC is also widely used for "system development life cycle." All usage of "SDLC" in this document is referencing software, not systems.

capabilities—for example, constant automated testing of software and software components to identify vulnerabilities, breaches of integrity, and other security issues.

2 Scenarios

The use case scenarios we are considering for this project are described below.

Scenario 1: Free and Open Source Software (FOSS) Development

This scenario involves a small FOSS community that wants to improve the security of their software. The FOSS community is all volunteer-based. They also want to provide better security transparency for others who want to use the software, including provenance information and mechanisms for confirming software integrity. This community already uses a cloud-based, publicly accessible version control repository for its software development, packaging, and distribution, plus patch management, configuration management, and other ongoing maintenance efforts. The software itself relies on multiple open source components from other communities.

Scenario 2: Closed Source Software Development

This scenario involves a medium- or large-size organization that has an existing cloud-based application for its global customers. The organization is actively developing, maintaining, and supporting the application, which utilizes multiple closed source and open source components. The application's production environment is in the public cloud and is microservices-based. The development and build environments, version control systems, code repositories, and other parts of the toolchain are spread across private clouds and Software-as-a-Service (SaaS)-hosted applications. In this scenario, the organization wants to ensure its DevSecOps approach addresses all applicable practices in the SSDF for its cloud environments, as well as generates artifacts to support and inform its self-attestation and declaration to conformance to applicable NIST and industry-recommended practices for secure software development and cybersecurity supply chain risk management.

For each scenario, we will perform one or more build implementations. Each build implementation will be significantly different from the others, such as using different technology stacks and programming languages. Each build implementation will rely on automation to the extent feasible, such as using existing capabilities or adding automated features into existing platforms and tools. Also, each build implementation will address security throughout the entire software development life cycle, to include the security of developer, integration, build, deployment, and distribution systems.

3 High-Level Architecture

Component List

The high-level architecture of the development and hosting environments may include, but is not limited to, the following components:

- Developer endpoints, including PCs (desktops or laptops) and virtual environments, both PC-based and cloud-based
- Network/infrastructure devices
- Services and applications, both on-premises and cloud-based
    - Toolchains and their tools (build tools, packaging tools, repositories, etc.)

    - Vulnerability management (patch and configuration)
    - Version control software and services
    - Software security review, analysis, and testing tools (e.g., static and dynamic code analyzers, fuzzers, just-in-time secure coding training for developers)
    - Secure software design tools (e.g., threat modeling tools)
- Build systems (test, integration, production)
- Distribution/delivery systems
- Production systems that host apps
- Hardware-enabled security capabilities for protecting private keys

Desired Security Capabilities

This project seeks to develop reference designs and implementations using commercially available closed source technology and open source technology that meet the following characteristics:

- Security practices as presented in Table 1 are applied throughout the entire software development lifecycle.
- Automation is used whenever feasible.

4 Relevant Standards and Guidance

The following resources and references provide additional information that could be leveraged to help develop this solution:

NIST Frameworks

- Framework for Improving Critical Infrastructure Cybersecurity, Version 1.1
- Risk Management Framework (RMF) Overview
- Secure Software Development Framework (SSDF) Version 1.1
- Workforce Framework for Cybersecurity (NICE Framework)

NIST Technology Projects

- Hardware Roots of Trust
- National Checklist Program
- Online Informative References (OLIR)
- Open Security Controls Assessment Language
- Security Content Automation Protocol (SCAP)
- Software Assurance Reference Dataset (SARD)

NIST Technology Guidelines

- Application Container Security Guide (SP 800-190)
- Building Secure Microservices-based Applications Using Service-Mesh Architecture (SP 800-204A)

- Cybersecurity Supply Chain Risk Management Practices for Systems and Organizations (SP 800-161 Rev. 1)
- Developing Cyber-Resilient Systems: A Systems Security Engineering Approach (SP 800-160 Vol. 2 Rev. 1)
- Guide to Enterprise Patch Management Planning: Preventive Maintenance for Technology (SP 800-40 Rev. 4)
- Guide to Security for Full Virtualization Technologies (SP 800-125)
- Hardware-Enabled Security: Enabling a Layered Approach to Platform Security for Cloud and Edge Computing Use Cases (IR 8320)
- Secure Virtual Network Configuration for Virtual Machine (VM) Protection (SP 800-125B)
- Security Recommendations for Server-based Hypervisor Platforms (SP 800-125A Rev. 1)
- Security Strategies for Microservices-based Application Systems (SP 800-204)
- Systems Security Engineering: Considerations for a Multidisciplinary Approach in the Engineering of Trustworthy Secure Systems (SP 800-160 Vol. 1)
- Zero Trust Architecture (SP 800-207)

Government, Industry, Academia, and Community Guidance and Practices

- BSA | The Software Alliance
- Carnegie Mellon University (CMU) Software Engineering Institute (SEI) DevSecOps Blog and DevSecOps Platform Independent Model (PIM) Portal Introduction
- Center for Internet Security (CIS) Benchmarks
- Cloud Native Computing Foundation (CNCF) Software Supply Chain Best Practices and The Secure Software Factory: A reference architecture to securing the software supply chain
- Cloud Security Alliance (CSA) DevSecOps Working Group
- Consortium for Information & Software Quality (CISQ) Standards to Automate Software Measurement
- Continuous Delivery (CD) Foundation
- Cybersecurity & Information Systems Information Analysis Center (CSIAC)
- Defense Information Systems Agency (DISA) Security Technical Implementation Guides (STIGs)
- Department of Defense (DoD) Enterprise DevSecOps Initiative
- General Services Administration (GSA) Tech Guides on DevSecOps
- Google Cloud, DevOps Research & Assessment (DORA)'s State of DevOps research program
- Michael Scovetta in collaboration with the Open Source Security Coalition, *Threats, Risks, and Mitigations in the Open Source Ecosystem*

- Microsoft, Open Source Software (OSS) Secure Supply Chain (SSC) Framework
- Microsoft and Sogeti, Securing Enterprise DevOps Environments
- Office of Management and Budget (OMB) Memorandum M-22-18, Enhancing the Security of the Software Supply Chain through Secure Software Development Practices
- Open Source Security Foundation (OpenSSF) resources, including:
    - The Alpha-Omega Project
    - Best Practices for Open Source Developers
    - Existing Guidelines for Developing and Distributing Secure Software
    - Guide to Security Tools
    - One-page Guide for Developing More Secure Software
    - Open Source Security Metrics
    - OpenSSF Best Practices Badge Program
    - Package Manager Best Practices
    - Security Reviews (of open source software)
    - Security Scorecards – Security health metrics for Open Source
    - sigstore
    - SLSA (Supply-chain Levels for Software Artifacts)
    - Supply Chain Integrity WG
    - Vulnerability Disclosures
    - WG Securing Critical Projects
- Papers and presentations from the International Workshop on Secure Software Engineering in DevOps and Agile Development
- Platform One, Big Bang
- Software Assurance Forum for Excellence in Code (SAFECode) publications on secure software development, including *Managing Security Risks Inherent in the Use of Third-Party Components*
- Supply Chain Integrity, Transparency, and Trust (SCITT)
- U.S. Open-Source Software Security Initiative Workshop and Recommendations from the Workshop on Open-Source Software Security Initiative, by Angelos Keromytis, Georgia Institute of Technology

5 Security Control Map

Table 1 maps the characteristics of the closed source and open source products that the NCCoE will apply to this cybersecurity challenge, as represented by SSDF practices and tasks, to the applicable standards and recommended practices described in the Framework for Improving Critical Infrastructure Cybersecurity, SP 800-53, SP 800-161, and Executive Order (EO) 14028.

The mappings indicate how performing SSDF practices and tasks can help satisfy elements of these other publications. This exercise is meant to demonstrate the real-world applicability of standards and best practices but does not imply that products with these characteristics will meet an industry's requirements for regulatory approval or accreditation.

Table 1 uses the following abbreviations for mapped publications:

- **EO14028**: *EO 14028, Executive Order on Improving the Nation's Cybersecurity* [6]
- **NISTCSF**: *NIST Cybersecurity Framework (Framework for Improving Critical Infrastructure Cybersecurity)* [3]
- **SP80053**: SP 800-53 Revision 5, *Security and Privacy Controls for Information Systems and Organizations* [7]
- **SP800161**: SP 800-161 Revision 1, *Cybersecurity Supply Chain Risk Management Practices for Systems and Organizations* [1]

Table 1: Security Control Map

SSDF Practices	SSDF Tasks	References
Define Security Requirements for Software Development (PO.1): Ensure that security requirements for software development are known at all times so that they can be taken into account throughout the SDLC and duplication of effort can be minimized because the requirements information can be collected once and shared. This includes 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).	**PO.1.1**: Identify and document all security requirements for the organization's software development infrastructures and processes, and maintain the requirements over time.	**EO14028**: 4e(ix) **NISTCSF**: ID.GV-3 **SP80053**: SA-1, SA-8, SA-15, SR-3 **SP800161**: SA-1, SA-8, SA-15, SR-3
	PO.1.2: Identify and document all security requirements for organization-developed software to meet, and maintain the requirements over time.	**EO14028**: 4e(ix) **NISTCSF**: ID.GV-3 **SP80053**: SA-8, SA-8(3), SA-15, SR-3 **SP800161**: SA-8, SA-15, SR-3
	PO.1.3: Communicate requirements to all third parties who will provide commercial software components to the organization for reuse by the organization's own software. [Formerly PW.3.1]	**EO14028**: 4e(vi), 4e(ix) **NISTCSF**: ID.SC-3 **SP80053**: SA-4, SA-9, SA-10, SA-10(1), SA-15, SR-3, SR-4, SR-5 **SP800161**: SA-4, SA-9, SA-9(1), SA-9(3), SA-10, SA-10(1), SA-15, SR-3, SR-4, SR-5
Implement Roles and Responsibilities (PO.2): Ensure that everyone inside and outside of the organization involved in the SDLC is prepared to perform their SDLC-related roles and responsibilities throughout the SDLC.	**PO.2.1**: Create new roles and alter responsibilities for existing roles as needed to encompass all parts of the SDLC. Periodically review and maintain the defined roles and responsibilities, updating them as needed.	**EO14028**: 4e(ix) **NISTCSF**: ID.AM-6, ID.GV-2 **SP80053**: SA-3 **SP800161**: SA-3
	PO.2.2: Provide role-based training for all personnel with responsibilities that contribute to secure development. Periodically review personnel proficiency and role-based training, and update the training as needed.	**EO14028**: 4e(ix) **NISTCSF**: PR.AT **SP80053**: SA-8 **SP800161**: SA-8
	PO.2.3: Obtain upper management or authorizing official commitment to secure development, and convey that commitment to all with development-related roles and responsibilities.	**EO14028**: 4e(ix) **NISTCSF**: ID.RM-1, ID.SC-1
Implement Supporting Toolchains (PO.3): Use automation to reduce human effort and improve the accuracy, reproducibility, usability, and comprehensiveness of security practices throughout the SDLC, as well as provide a way to document and demonstrate the use of these practices. Toolchains and tools may be used at	**PO.3.1**: Specify which tools or tool types must or should be included in each toolchain to mitigate identified risks, as well as how the toolchain components are to be integrated with each other.	**EO14028**: 4e(iii), 4e(ix) **SP80053**: SA-15 **SP800161**: SA-15
	PO.3.2: Follow recommended security practices to deploy, operate, and maintain tools and toolchains.	**EO14028**: 4e(i)(F), 4e(ii), 4e(iii), 4e(v), 4e(vi), 4e(ix) **SP80053**: SA-15 **SP800161**: SA-15

SSDF Practices	SSDF Tasks	References
different levels of the organization, such as organization-wide or project-specific, and may address a particular part of the SDLC, like a build pipeline.	**PO.3.3**: Configure tools to generate artifacts of their support of secure software development practices as defined by the organization.	**EO14028**: 4e(i)(F), 4e(ii), 4e(v), 4e(ix) **SP80053**: SA-15 **SP800161**: SA-15
Define and Use Criteria for Software Security Checks (PO.4): Help ensure that the software resulting from the SDLC meets the organization's expectations by defining and using criteria for checking the software's security during development.	**PO.4.1**: Define criteria for software security checks and track throughout the SDLC.	**EO14028**: 4e(iv), 4e(v), 4e(ix) **SP80053**: SA-15, SA-15(1) **SP800161**: SA-15, SA-15(1)
	PO.4.2: Implement processes, mechanisms, etc. to gather and safeguard the necessary information in support of the criteria.	**EO14028**: 4e(iv), 4e(v), 4e(ix) **SP80053**: SA-15, SA-15(1), SA-15(11) **SP800161**: SA-15, SA-15(1), SA-15(11)
Implement and Maintain Secure Environments for Software Development (PO.5): Ensure that all components of the environments for software development are strongly protected from internal and external threats to prevent compromises of the environments or the software being developed or maintained within them. Examples of environments for software development include development, build, test, and distribution environments.	**PO.5.1**: Separate and protect each environment involved in software development.	**EO14028**: 4e(i)(A), 4e(i)(B), 4e(i)(C), 4e(i)(D), 4e(i)(F), 4e(ii), 4e(iii), 4e(v), 4e(vi), 4e(ix) **NISTCSF**: PR.AC-5, PR.DS-7 **SP80053**: SA-3(1), SA-8, SA-15 **SP800161**: SA-3, SA-8, SA-15
	PO.5.2: Secure and harden development endpoints (i.e., endpoints for software designers, developers, testers, builders, etc.) to perform development-related tasks using a risk-based approach.	**EO14028**: 4e(i)(C), 4e(i)(E), 4e(i)(F), 4e(ii), 4e(iii), 4e(v), 4e(vi), 4e(ix) **NISTCSF**: PR.AC-4, PR.AC-7, PR.IP-1, PR.IP-3, PR.IP-12, PR.PT-1, PR.PT-3, DE.CM **SP80053**: SA-15 **SP800161**: SA-15
Protect All Forms of Code from Unauthorized Access and Tampering (PS.1): Help prevent unauthorized changes to code, both inadvertent and intentional, which could circumvent or negate the intended security characteristics of the software. For code that is not intended to be publicly accessible, this helps prevent theft of the software and may make it more difficult or time-consuming for attackers to find vulnerabilities in the software.	**PS.1.1**: Store all forms of code – including source code, executable code, and configuration-as-code – based on the principle of least privilege so that only authorized personnel, tools, services, etc. have access.	**EO14028**: 4e(iii), 4e(iv), 4e(ix) **NISTCSF**: PR.AC-4, PR.DS-6, PR.IP-3 **SP80053**: SA-10 **SP800161**: SA-8, SA-10
Provide a Mechanism for Verifying Software Release Integrity (PS.2): Help software acquirers ensure that the software they acquire is legitimate and has not been tampered with.	**PS.2.1**: Make software integrity verification information available to software acquirers.	**EO14028**: 4e(iii), 4e(ix), 4e(x) **NISTCSF**: PR.DS-6 **SP80053**: SA-8 **SP800161**: SA-8

SSDF Practices	SSDF Tasks	References
Archive and Protect Each Software Release (PS.3): Preserve software releases in order to help identify, analyze, and eliminate vulnerabilities discovered in the software after release.	**PS.3.1**: Securely archive the necessary files and supporting data (e.g., integrity verification information, provenance data) to be retained for each software release.	**EO14028**: 4e(iii), 4e(vi), 4e(ix), 4e(x) **NISTCSF**: PR.IP-4 **SP80053**: SA-10, SA-15, SA-15(11), SR-4 **SP800161**: SA-8, SA-10, SA-15(11), SR-4
	PS.3.2: Collect, safeguard, maintain, and share provenance data for all components of each software release (e.g., in a software bill of materials [SBOM]).	**EO14028**: 4e(vi), 4e(vii), 4e(ix), 4e(x) **SP80053**: SA-8, SR-3, SR-4 **SP800161**: SA-8, SR-3, SR-4
Design Software to Meet Security Requirements and Mitigate Security Risks (PW.1): Identify and evaluate the security requirements for the software; determine what security risks the software is likely to face during operation and how the software's design and architecture should mitigate those risks; and justify any cases where risk-based analysis indicates that security requirements should be relaxed or waived. Addressing security requirements and risks during software design (secure by design) is key for improving software security and also helps improve development efficiency.	**PW.1.1**: Use forms of risk modeling – such as threat modeling, attack modeling, or attack surface mapping – to help assess the security risk for the software.	**EO14028**: 4e(ix) **NISTCSF**: ID.RA **SP80053**: SA-8, SA-11(2), SA-11(6), SA-15(5) **SP800161**: SA-8, SA-11(2), SA-11(6), SA-15(5)
	PW.1.2: Track and maintain the software's security requirements, risks, and design decisions.	**EO14028**: 4e(v), 4e(ix) **SP80053**: SA-8, SA-10, SA-17 **SP800161**: SA-8, SA-17
	PW.1.3: Where appropriate, build in support for using standardized security features and services (e.g., enabling software to integrate with existing log management, identity management, access control, and vulnerability management systems) instead of creating proprietary implementations of security features and services. [Formerly PW.4.3]	**EO14028**: 4e(ix)
Review the Software Design to Verify Compliance with Security Requirements and Risk Information (PW.2): Help ensure that the software will meet the security requirements and satisfactorily address the identified risk information.	**PW.2.1**: Have 1) a qualified person (or people) who were not involved with the design and/or 2) automated processes instantiated in the toolchain review the software design to confirm and enforce that it meets all of the security requirements and satisfactorily addresses the identified risk information.	**EO14028**: 4e(iv), 4e(v), 4e(ix)
Verify Third-Party Software Complies with Security Requirements (PW.3)*: Moved to PW.4*	***PW.3.1****: Moved to PO.1.3*	
	PW.3.2*: Moved to PW.4.4*	
Reuse Existing, Well-Secured Software When Feasible Instead of Duplicating Functionality (PW.4): Lower the costs of software development, expedite software development, and decrease the likelihood of introducing additional security vulnerabilities into the	**PW.4.1**: Acquire and maintain well-secured software components (e.g., software libraries, modules, middleware, frameworks) from commercial, open-source, and other third-party developers for use by the organization's software.	**EO14028**: 4e(iii), 4e(vi), 4e(ix), 4e(x) **NISTCSF**: ID.SC-2 **SP80053**: SA-4, SA-5, SA-8(3), SA-10(6), SR-3, SR-4 **SP800161**: SA-4, SA-5, SA-8(3), SA-10(6), SR-3, SR-4

SSDF Practices	SSDF Tasks	References
software by reusing software modules and services that have already had their security posture checked. This is particularly important for software that implements security functionality, such as cryptographic modules and protocols.	**PW.4.2**: Create and maintain well-secured software components in-house following SDLC processes to meet common internal software development needs that cannot be better met by third-party software components.	**EO14028**: 4e(ix) **SP80053**: SA-8(3) **SP800161**: SA-8(3)
	***PW.4.3**: Moved to PW.1.3*	
	PW.4.4: Verify that acquired commercial, open-source, and all other third-party software components comply with the requirements, as defined by the organization, throughout their life cycles.	**EO14028**: 4e(iii), 4e(iv), 4e(vi), 4e(ix), 4e(x) **NISTCSF**: ID.SC-4, PR.DS-6 **SP80053**: SA-9, SR-3, SR-4, SR-4(3), SR-4(4) **SP800161**: SA-4, SA-8, SA-9, SA-9(3), SR-3, SR-4, SR-4(3), SR-4(4)
	***PW.4.5**: Moved to PW.4.1 and PW.4.4*	
Create Source Code by Adhering to Secure Coding Practices (PW.5): Decrease the number of security vulnerabilities in the software, and reduce costs by minimizing vulnerabilities introduced during source code creation that meet or exceed organization-defined vulnerability severity criteria.	**PW.5.1**: Follow all secure coding practices that are appropriate to the development languages and environment to meet the organization's requirements.	**EO14028**: 4e(iv), 4e(ix)
	***PW.5.2**: Moved to PW.5.1 as example*	
Configure the Compilation, Interpreter, and Build Processes to Improve Executable Security (PW.6): Decrease the number of security vulnerabilities in the software and reduce costs by eliminating vulnerabilities before testing occurs.	**PW.6.1**: Use compiler, interpreter, and build tools that offer features to improve executable security.	**EO14028**: 4e(iv), 4e(ix) **SP80053**: SA-15 **SP800161**: SA-15
	PW.6.2: Determine which compiler, interpreter, and build tool features should be used and how each should be configured, then implement and use the approved configurations.	**EO14028**: 4e(iv), 4e(ix) **SP80053**: SA-15, SR-9 **SP800161**: SA-15, SR-9
Review and/or Analyze Human-Readable Code to Identify Vulnerabilities and Verify Compliance with Security Requirements (PW.7): Help identify vulnerabilities so that 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, scripts, and any other form of code that an organization deems human-	**PW.7.1**: Determine whether code *review* (a person looks directly at the code to find issues) and/or code *analysis* (tools are used to find issues in code, either in a fully automated way or in conjunction with a person) should be used, as defined by the organization.	**EO14028**: 4e(iv), 4e(ix) **SP80053**: SA-11 **SP800161**: SA-11
	PW.7.2: Perform the code review and/or code analysis based on the organization's secure coding standards, and record and triage all discovered issues and recommended remediations in the development team's workflow or issue tracking system.	**EO14028**: 4e(iv), 4e(v), 4e(ix) **SP80053**: SA-11, SA-11(1), SA-11(4), SA-15(7) **SP800161**: SA-11, SA-11(1), SA-11(4), SA-15(7)

SSDF Practices	SSDF Tasks	References
readable.		
Test Executable Code to Identify Vulnerabilities and Verify Compliance with Security Requirements (PW.8): Help identify vulnerabilities so that they can be corrected before the software is released in order to prevent exploitation. Using automated methods lowers the effort and resources needed to detect vulnerabilities and improves traceability and repeatability. Executable code includes binaries, directly executed bytecode and source code, and any other form of code that an organization deems executable.	**PW.8.1**: Determine whether executable code testing should be performed to find vulnerabilities not identified by previous reviews, analysis, or testing and, if so, which types of testing should be used.	**EO14028**: 4e(ix) **SP80053**: SA-11 **SP800161**: SA-11
	PW.8.2: Scope the testing, design the tests, perform the testing, and document the results, including recording and triaging all discovered issues and recommended remediations in the development team's workflow or issue tracking system.	**EO14028**: 4e(iv), 4e(v), 4e(ix) **SP80053:** SA-11, SA-11(5), SA-11(8), SA-15(7) **SP800161**: SA-11, SA-11(5), SA-11(8), SA-15(7)
Configure Software to Have Secure Settings by Default (PW.9): Help improve the security of the software at the time of installation to reduce the likelihood of the software being deployed with weak security settings, putting it at greater risk of compromise.	**PW.9.1**: Define a secure baseline by determining how to configure each setting that has an effect on security or a security-related setting so that the default settings are secure and do not weaken the security functions provided by the platform, network infrastructure, or services.	**EO14028**: 4e(iv), 4e(ix)
	PW.9.2: Implement the default settings (or groups of default settings, if applicable), and document each setting for software administrators.	**EO14028**: 4e(iv), 4e(ix) **SP80053**: SA-5, SA-8(23) **SP800161**: SA-5, SA-8(23)
Identify and Confirm Vulnerabilities on an Ongoing Basis (RV.1): Help ensure that vulnerabilities are identified more quickly so that they can be remediated more quickly in accordance with risk, reducing the window of opportunity for attackers.	**RV.1.1**: Gather information from software acquirers, users, and public sources on potential vulnerabilities in the software and third-party components that the software uses, and investigate all credible reports.	**EO14028**: 4e(iv), 4e(vi), 4e(viii), 4e(ix) **SP80053**: SA-10, SR-3, SR-4 **SP800161**: SA-10, SR-3, SR-4
	RV.1.2: Review, analyze, and/or test the software's code to identify or confirm the presence of previously undetected vulnerabilities.	**EO14028**: 4e(iv), 4e(vi), 4e(viii), 4e(ix) **SP80053**: SA-11 **SP800161**: SA-11
	RV.1.3: Have a policy that addresses vulnerability disclosure and remediation, and implement the roles, responsibilities, and processes needed to support that policy.	**EO14028**: 4e(viii), 4e(ix) **SP80053**: SA-15(10) **SP800161**: SA-15(10)
Assess, Prioritize, and Remediate Vulnerabilities (RV.2): Help ensure that vulnerabilities are remediated in accordance with	**RV.2.1**: Analyze each vulnerability to gather sufficient information about risk to plan its remediation or other risk response.	**EO14028**: 4e(iv), 4e(viii), 4e(ix) **SP80053**: SA-10, SA-15(7) **SP800161**: SA-15(7)

SSDF Practices	SSDF Tasks	References
risk to reduce the window of opportunity for attackers.	**RV.2.2**: Plan and implement risk responses for vulnerabilities.	**EO14028**: 4e(iv), 4e(vi), 4e(viii), 4e(ix) **SP80053**: SA-5, SA-10, SA-11, SA-15(7) **SP800161**: SA-5, SA-8, SA-10, SA-11, SA-15(7)
Analyze Vulnerabilities to Identify Their Root Causes (RV.3): Help reduce the frequency of vulnerabilities in the future.	**RV.3.1**: Analyze identified vulnerabilities to determine their root causes.	**EO14028**: 4e(ix)
	RV.3.2: Analyze the root causes over time to identify patterns, such as a particular secure coding practice not being followed consistently.	**EO14028**: 4e(ix)
	RV.3.3: Review the software for similar vulnerabilities to eradicate a class of vulnerabilities, and proactively fix them rather than waiting for external reports.	**EO14028**: 4e(iv), 4e(viii), 4e(ix) **SP80053**: SA-11 **SP800161**: SA-11
	RV.3.4: Review the SDLC process, and update it if appropriate to prevent (or reduce the likelihood of) the root cause recurring in updates to the software or in new software that is created.	**EO14028**: 4e(ix) **SP80053**: SA-15 **SP800161**: SA-15

Appendix A References

[1] J. Boyens et al., *Cybersecurity Supply Chain Risk Management Practices for Systems and Organizations*, National Institute of Standards and Technology (NIST) Special Publication (SP) 800-161 Revision 1, Gaithersburg, Md., May 2022, 326 pp. Available: https://doi.org/10.6028/NIST.SP.800-161r1.

[2] M. Souppaya et al., *Secure Software Development Framework (SSDF) Version 1.1: Recommendations for Mitigating the Risk of Software Vulnerabilities*, National Institute of Standards and Technology (NIST) Special Publication (SP) 800-218, Gaithersburg, Md., February 2022, 36 pp. Available: https://doi.org/10.6028/NIST.SP.800-218.

[3] NIST, Framework for Improving Critical Infrastructure Cybersecurity, Version 1.1, 2018. https://doi.org/10.6028/NIST.CSWP.04162018.

[4] M. Souppaya et al., *Application Container Security Guide*, National Institute of Standards and Technology (NIST) Special Publication (SP) 800-190, Gaithersburg, Md., September 2017, 63 pp. Available: https://doi.org/10.6028/NIST.SP.800-190.

[5] E. LeMay et al., *The Common Misuse Scoring System (CMSS): Metrics for Software Feature Misuse Vulnerabilities*, National Institute of Standards and Technology (NIST) Internal Report (IR) 7864, Gaithersburg, Md., July 2012, 39 pp. Available: https://doi.org/10.6028/NIST.IR.7864.

[6] *Executive Order on Improving the Nation's Cybersecurity*, Executive Order (EO) 14028, May 12, 2021. Available: https://www.whitehouse.gov/briefing-room/presidential-actions/2021/05/12/executive-order-on-improving-the-nations-cybersecurity/.

[7] Joint Task Force, *Security and Privacy Controls for Information Systems and Organizations*, National Institute of Standards and Technology (NIST) Special Publication (SP) 800-53 Revision 5, Gaithersburg, Md., September 2020, 492 pp. Available: https://doi.org/10.6028/NIST.SP.800-53r5.

Appendix B Acronyms and Abbreviations

ATO	Authorization to Operate
CD	Continuous Delivery
CI/CD	Continuous Integration/Continuous Delivery
CIS	Center for Internet Security
CISQ	Consortium for Information & Software Quality
CMU	Carnegie Mellon University
CNCF	Cloud Native Computing Foundation
CSA	Cloud Security Alliance
C-SCRM	Cybersecurity Supply Chain Risk Management
CSIAC	Cyber Security & Information Systems Information Analysis Center
DevOps	Software Development and IT Operations
DevSecOps	Software Development, Security, and IT Operations
DISA	Defense Information Systems Agency
DoD	Department of Defense
DORA	(Google Cloud) DevOps Research & Assessment
EO	Executive Order
FOSS	Free and Open Source Software
GSA	General Services Administration
IoT	Internet of Things
IT	Information Technology
NCCoE	National Cybersecurity Center of Excellence
NICE	National Initiative for Cybersecurity Education
NIST	National Institute of Standards and Technology
OLIR	Online Informative References
OMB	Office of Management and Budget
OpenSSF	Open Source Security Foundation
OSS	Open Source Software
OT	Operational Technology
PC	Personal Computer
PIM	Platform Independent Model
RMF	Risk Management Framework

SaaS	Software as a Service
SAFECode	Software Assurance Forum for Excellence in Code
SARD	Software Assurance Reference Dataset
SBOM	Software Bill of Materials
SCAP	Security Content Automation Protocol
SCITT	Supply Chain Integrity, Transparency, and Trust
SDLC	Software Development Life Cycle
SEI	Software Engineering Institute
SLSA	Supply-Chain Levels for Software Artifacts
SP	Special Publication
SSC	Secure Supply Chain
SSDF	Secure Software Development Framework
STIG	Security Technical Implementation Guide
VM	Virtual Machine