Draft NIST Special Publication 800-218

Secure Software Development Framework (SSDF) Version 1.1:

Recommendations for Mitigating the Risk of Software Vulnerabilities

Murugiah Souppaya
Karen Scarfone
Donna Dodson

This publication is available free of charge from:
https://doi.org/10.6028/NIST.SP.800-218-draft

National Institute of Standards and Technology
U.S. Department of Commerce

Draft NIST Special Publication 800-218

Secure Software Development Framework (SSDF) Version 1.1:

Recommendations for Mitigating the Risk of Software Vulnerabilities

Murugiah Souppaya
Computer Security Division
Information Technology Laboratory

Karen Scarfone
Scarfone Cybersecurity
Clifton, VA

Donna Dodson*
** Former NIST employee; all work for this publication was done while at NIST.*

This publication is available free of charge from:
https://doi.org/10.6028/NIST.SP.800-218-draft

September 2021

U.S. Department of Commerce
Gina M. Raimondo, Secretary

National Institute of Standards and Technology
James K. Olthoff, Performing the Non-Exclusive Functions and Duties of the Under Secretary of Commerce for Standards and Technology & Director, National Institute of Standards and Technology

Authority

This publication has been developed by NIST in accordance with its statutory responsibilities under the Federal Information Security Modernization Act (FISMA) of 2014, 44 U.S.C. § 3551 *et seq.*, Public Law (P.L.) 113-283. NIST is responsible for developing information security standards and guidelines, including minimum requirements for federal information systems, but such standards and guidelines shall not apply to national security systems without the express approval of appropriate federal officials exercising policy authority over such systems. This guideline is consistent with the requirements of the Office of Management and Budget (OMB) Circular A-130.

Nothing in this publication should be taken to contradict the standards and guidelines made mandatory and binding on federal agencies by the Secretary of Commerce under statutory authority. Nor should these guidelines be interpreted as altering or superseding the existing authorities of the Secretary of Commerce, Director of the OMB, or any other federal official. This publication may be used by nongovernmental organizations on a voluntary basis and is not subject to copyright in the United States. Attribution would, however, be appreciated by NIST.

National Institute of Standards and Technology Special Publication 800-218
Natl. Inst. Stand. Technol. Spec. Publ. 800-218, 31 pages (September 2021)
CODEN: NSPUE2

This publication is available free of charge from:
https://doi.org/10.6028/NIST.SP.800-218-draft

Public comment period: *September 30, 2021* through *November 5, 2021*

National Institute of Standards and Technology
Attn: Computer Security Division, Information Technology Laboratory
100 Bureau Drive (Mail Stop 8930) Gaithersburg, MD 20899-8930
Email: ssdf@nist.gov

Reports on Computer Systems Technology

The Information Technology Laboratory (ITL) at the National Institute of Standards and Technology (NIST) promotes the U.S. economy and public welfare by providing technical leadership for the Nation's measurement and standards infrastructure. ITL develops tests, test methods, reference data, proof of concept implementations, and technical analyses to advance the development and productive use of information technology. ITL's responsibilities include the development of management, administrative, technical, and physical standards and guidelines for the cost-effective security and privacy of other than national security-related information in federal information systems. The Special Publication 800-series reports on ITL's research, guidelines, and outreach efforts in information system security, and its collaborative activities with industry, government, and academic organizations.

Abstract

Few software development life cycle (SDLC) models explicitly address software security in detail, so secure software development practices usually need to be added to each SDLC model to ensure that the software being developed is well-secured. This document recommends the Secure Software Development Framework (SSDF) – a core set of high-level secure software development practices that can be integrated into each SDLC implementation. Following these practices should help software producers reduce the number of vulnerabilities in released software, mitigate the potential impact of the exploitation of undetected or unaddressed vulnerabilities, and address the root causes of vulnerabilities to prevent future recurrences. Because the framework provides a common vocabulary for secure software development, software purchasers and consumers can also use it to foster communications with suppliers in acquisition processes and other management activities.

Keywords

secure software development; Secure Software Development Framework (SSDF); secure software development practices; software acquisition; software development; software development life cycle (SDLC); software security.

Trademark Information

Acknowledgments

The authors thank all of the organizations and individuals who provided input for this update to the SSDF. In response to Section 4 of Executive Order (EO) 14028 on "Improving the Nation's Cybersecurity," NIST held a June 2021 workshop and received over 150 position papers, many of which suggested secure software development practices, tasks, examples of implementations, and references for consideration for this SSDF update. The authors appreciate all of those suggestions, as well as the inputs from those who spoke at the workshop or attended the workshop and shared their thoughts during or after the workshop.

The authors also wish to thank all of the individuals and organizations who provided comments on drafts of the original version of the SSDF, including the Administrative Offices of the U.S. Courts, The Aerospace Corporation, BSA | The Software Alliance, Capitis Solutions, the Consortium for Information & Software Quality (CISQ), HackerOne, Honeycomb Secure Systems, iNovex, Ishpi Information Technologies, the Information Security and Privacy Advisory Board (ISPAB), Juniper Networks, Medical Imaging & Technology Alliance (MITA), Microsoft, Naval Sea Systems Command (NAVSEA), the National Institute of Standards and Technology (NIST), Northrop Grumman, the Office of the Undersecretary of Defense for Research and Engineering, Red Hat, the Software Assurance Forum for Excellence in Code (SAFECode), and the Software Engineering Institute (SEI).

Audience

There are two primary audiences for this document. The first is software producers (e.g., commercial-off-the-shelf [COTS] product vendors, government-off-the-shelf [GOTS] software developers, custom software developers) regardless of size, sector, or level of maturity. The second is software purchasers and consumers, both federal agencies and other organizations. Readers of this document are not expected to be experts in secure software development in order to understand it, but such expertise is required to implement its recommended practices.

Personnel within the following Workforce Categories and Specialty Areas from the National Initiative for Cybersecurity Education (NICE) Cybersecurity Workforce Framework [SP800181] are most likely to find this publication of interest:

- Securely Provision (SP): Risk Management (RSK), Software Development (DEV), Systems Requirements Planning (SRP), Test and Evaluation (TST), Systems Development (SYS)
- Operate and Maintain (OM): Systems Analysis (ANA)
- Oversee and Govern (OV): Training, Education, and Awareness (TEA); Cybersecurity Management (MGT); Executive Cyber Leadership (EXL); Program/Project Management (PMA) and Acquisition
- Protect and Defend (PR): Incident Response (CIR), Vulnerability Assessment and Management (VAM)
- Analyze (AN): Threat Analysis (TWA), Exploitation Analysis (EXP)

Note to Reviewers

The authors welcome feedback on any part of this document but are particularly interested in the following:

- Do the SSDF practices, tasks, and implementation examples fit well into your current software development practices? Are there any conflicts or gaps that the SSDF should address?
- Should the SSDF practices and tasks involving software integration, building, and delivery be split so that integration is separate from building and delivery?
- What types of artifacts and evidence can be captured, documented, and shared publicly as byproducts of implementing the secure software development practices? Are there examples you can share?

If you are from a standards developing organization or another organization that has produced a set of secure practices, and you would like to map your secure software development standard or guidance to the SSDF, please contact the authors at ssdf@nist.gov. They would like to introduce you to the National Online Informative References Program (OLIR) so that you can submit your mapping there to augment the existing set of informative references.

Call for Patent Claims

This public review includes a call for information on essential patent claims (claims whose use would be required for compliance with the guidance or requirements in this Information Technology Laboratory (ITL) draft publication). Such guidance and/or requirements may be directly stated in this ITL Publication or by reference to another publication. This call also includes disclosure, where known, of the existence of pending U.S. or foreign patent applications relating to this ITL draft publication and of any relevant unexpired U.S. or foreign patents.

ITL may require from the patent holder, or a party authorized to make assurances on its behalf, in written or electronic form, either:

a) assurance in the form of a general disclaimer to the effect that such party does not hold and does not currently intend holding any essential patent claim(s); or

b) assurance that a license to such essential patent claim(s) will be made available to applicants desiring to utilize the license for the purpose of complying with the guidance or requirements in this ITL draft publication either:

   i. under reasonable terms and conditions that are demonstrably free of any unfair discrimination; or
   ii. without compensation and under reasonable terms and conditions that are demonstrably free of any unfair discrimination.

Such assurance shall indicate that the patent holder (or third party authorized to make assurances on its behalf) will include in any documents transferring ownership of patents subject to the assurance, provisions sufficient to ensure that the commitments in the assurance are binding on the transferee, and that the transferee will similarly include appropriate provisions in the event of future transfers with the goal of binding each successor-in-interest.

The assurance shall also indicate that it is intended to be binding on successors-in-interest regardless of whether such provisions are included in the relevant transfer documents.

Such statements should be addressed to: ssdf@nist.gov

Executive Summary

This document describes a set of fundamental, sound practices for secure software development called the Secure Software Development Framework (SSDF). Organizations should integrate the SSDF throughout their existing software development practices, express their secure software development requirements to third-party suppliers using SSDF conventions, and acquire software that meets the practices described in the SSDF. Using the SSDF helps organizations to meet the following secure software development recommendations:

- Organizations should ensure that their people, processes, and technology are prepared to perform secure software development.
- Organizations should protect all components of their software from tampering and unauthorized access.
- Organizations should produce well-secured software with minimal security vulnerabilities in its releases.
- Organizations should identify residual vulnerabilities in their software releases and respond appropriately to address those vulnerabilities and prevent similar ones from occurring in the future.

The SSDF does not prescribe exactly how to implement each practice. The focus is on the outcomes of the practices rather than on the tools, techniques, and mechanisms to do so. This means that the SSDF can be used by organizations in any sector or community, regardless of size or cybersecurity sophistication. It can be used for any type of software development, regardless of technology, platform, programming language, or operating environment.

The SSDF defines only a high-level subset of what organizations may need to do, so organizations should consult the references and other resources for additional information on implementing the practices. Not all practices are applicable to all use cases; organizations should adopt a risk-based approach to determine what practices are relevant, appropriate, and effective to mitigate the threats to their software development practices.

Organizations can communicate how they are meeting the clauses from section 4 of the President's Executive Order (EO) on "Improving the Nation's Cybersecurity (14028)" using the SSDF practices and tasks described in Appendix A.

Table of Contents

Executive Summary ..... **vi**
1 Introduction **1**
2 The Secure Software Development Framework **4**
References **17**
Appendix A— The SSDF and Executive Order 14028 ..... **20**
Appendix B— Acronyms **21**
Appendix C— Change Log **22**

List of Tables

Table 1: The Secure Software Development Framework (SSDF) Version 1.1 5
Table 2: SSDF Practices Corresponding to EO 14028 Clauses 20

1 Introduction

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 mitigate the potential impact of the exploitation of undetected or unaddressed vulnerabilities, and to address the root causes of vulnerabilities to prevent recurrences.

Most aspects of security can be addressed multiple times within an SDLC, but 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, also 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.

There are many existing documents on secure software development practices, including those listed in the References section. This document does not introduce new practices or define new terminology; instead, it describes a set of recommended high-level practices based on established standards, guidance, and secure software development practice documents. These practices, collectively called the Secure Software Development Framework (SSDF), are intended to help the target audiences achieve secure software development objectives. Many of the practices directly involve the software itself, while others indirectly involve it (e.g., securing the development environment).

Future work may expand on these recommendations, potentially covering topics such as how the SSDF may apply to and vary for particular software development methodologies and associated practices like DevOps and how an organization can transition from using just their current software development practices to also incorporating the practices specified by the SSDF. Future work will likely take the form of use cases so that the insights will be more readily applicable to various types of development environments.

This document identifies and recommends secure software development practices but does not prescribe exactly how to implement them. The focus is on the outcomes of the practices to be implemented rather than on the tools, techniques, and mechanisms used to do so. Advantages of specifying the practices at a high level include the following:

[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.

- Can be used by organizations in any sector or community, regardless of size or cybersecurity sophistication
- Can be applied to software developed to support information technology (IT), industrial control systems (ICS), cyber-physical systems (CPS), or the Internet of Things (IoT)
- Can be integrated into any existing software development workflow and automated toolchain; should not negatively affect organizations that already have robust, secure software development practices in place
- Makes the practices broadly applicable, not specific to particular technologies, platforms, programming languages, SDLC models, development environments, operating environments, tools, etc.
- Can help an organization document its secure software development practices today and define its future target practices as part of its continuous improvement process
- Can assist an organization currently using a classic software development model in transitioning its secure software development practices for use with a modern software development model (e.g., agile, DevOps)
- Can assist organizations that are procuring and using software to understand secure software development practices employed by their suppliers

This document also provides a common language to describe fundamental secure software development practices. This is similar to the approach taken by the *Framework for Improving Critical Infrastructure Cybersecurity*, also known as the NIST Cybersecurity Framework [NISTCSF].[2] Expertise in secure software development is not required to understand the practices. The common language helps facilitate communications about secure software practices among both internal and external organizational stakeholders, such as:

- Business owners, software developers, project managers and leads, cybersecurity professionals, and operations and platform engineers within an organization who need to clearly communicate with each other about secure software development
- Software purchasers and consumers, including both Federal Government agencies and other organizations, that want to define required or desired characteristics for software in their acquisition processes in order to have higher-quality software (particularly with fewer security vulnerabilities)[3]
- Software producers (e.g., commercial-off-the-shelf [COTS] product vendors, government-off-the-shelf [GOTS] software developers, software developers working within or on behalf of software consumer organizations, software testers/quality

[2] The SSDF practices may help support the NIST Cybersecurity Framework Functions, Categories, and Subcategories, but the SSDF practices do not map to them and are typically the responsibility of different parties. Developers can adopt SSDF practices, and the outcomes of their work could help organizations with their operational security in support of the Cybersecurity Framework.

[3] Future work may provide more practical guidance for software consumers on how they can leverage the SSDF in specific use cases.

assurance personnel) who want to integrate secure software development practices throughout their SDLCs, express their secure software practices to their customers, or define requirements for their suppliers

This document's practices are not based on the assumption that all organizations have the same security objectives and priorities. Rather, the recommendations reflect that each software producer may have unique security assumptions, and each software consumer may have unique security needs and requirements. While the aim is for each software producer to follow all applicable practices, the expectation is that the degree to which each practice is implemented and the formality of the implementation will vary based on the producer's security assumptions. The practices provide flexibility for implementers, but they are also clear to avoid leaving too much open to interpretation.

Although most of these practices are relevant to any software development effort, some are not. For example, if developing a particular piece of software does not involve using a compiler, there would be no need to follow a practice on configuring the compiler to improve executable security. Some practices are foundational, while others are more advanced and depend on certain foundational practices already being in place. Also, practices are not all equally important for all cases. Risk should be considered when deciding which practices to use and how much time and resources to devote to each practice.[4] The practices, tasks, and implementation examples are not prioritized. Finally, the frequency for performing recurring practices is not specified because the frequency appropriate for any particular situation depends on risk and other factors defined by the organization.

The responsibility for implementing the practices is distributed among different organizations based on the delivery of the software and services (e.g., on premises, infrastructure as a service, software as a service, platform as a service, container as a service, serverless). It follows a shared responsibility model involving the platform/service providers and the tenant who is consuming those platforms/services.

[4] Organizations seeking guidance on how to get started with secure software development can consult many publicly available references, such as "SDL That Won't Break the Bank" by Steve Lipner from SAFECode (https://i.blackhat.com/us-18/Thu-August-9/us-18-Lipner-SDL-For-The-Rest-Of-Us.pdf) and "Simplified Implementation of the Microsoft SDL" by Microsoft (https://www.microsoft.com/en-us/download/details.aspx?id=12379).

2 The Secure Software Development Framework

This document defines version 1.1 of the Secure Software Development Framework (SSDF), with fundamental, sound, and secure recommended practices based on established secure software development practice documents. The practices are organized into four groups:

- **Prepare the Organization (PO):** Organizations should ensure that their people, processes, and technology are prepared to perform secure software development at the organization level. Many organizations will find some PO practices to also be applicable to subsets of their secure software development, like individual development groups or projects.
- **Protect the Software (PS):** Organizations should protect all components of the software from tampering and unauthorized access.
- **Produce Well-Secured Software (PW):** Organizations should produce well-secured software with minimal security vulnerabilities in its releases.
- **Respond to Vulnerabilities (RV):** Organizations should identify residual vulnerabilities in software releases and respond appropriately to address those vulnerabilities and prevent similar ones from occurring in the future.

Each practice definition includes the following elements:

- **Practice:** The name of the practice and a unique identifier, followed by a brief explanation of what the practice is and why it is beneficial
- **Tasks:** One or more actions needed to accomplish a practice
- **Implementation Examples:** One or more examples of types of tools, processes, or other methods that could be used to help implement a task; not intended to imply that any example or combination of examples is required or that only the stated examples are feasible options
- **References:** Pointers to one or more established secure development practice documents and their mappings to a particular task; not all references will apply to all instances of software development

Table 1 defines the practices. They are only a **subset** of what an organization may need to do with the practices focused on helping organizations achieve secure software development objectives. The information in the table is space constrained, and much more information on each practice can be found in the references.

Note: The order of the practices and tasks in the table is not intended to imply the sequence of implementation or the relative importance of any practice or task.

Table 1: The Secure Software Development Framework (SSDF) Version 1.1

Practices	Tasks	Implementation Examples	References
Prepare the Organization (PO)			
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.	• Define policies for securing software development infrastructures and their components, including development endpoints, throughout the SDLC and maintaining that security. • Define policies for securing software development processes throughout the SDLC and maintaining that security, including open-source and other third-party software components utilized by software being developed. • Review all security requirements at least annually or sooner if there are new requirements from internal or external sources or if a major vulnerability incident has occurred. • Educate affected individuals on impending changes to requirements.	**BSAFSS:** SM.3, DE.1, IA.1, IA.2 **BSIMM**: CP1.1, CP1.3, SR1.1 **IEC62443**: SM-7, SM-9 **NISTCSF**: ID.GV-3 **OWASPASVS**: 1.1.1 **OWASPMASVS**: 1.10 **OWASPSAMM**: PC1-A, PC1-B, PC2-A **PCISSLC**: 2.1, 2.2 **SCFPSSD**: Planning the Implementation and Deployment of Secure Development Practices **SP80053**: SA-8, SA-15 **SP800160**: 3.1.2, 3.2.1, 3.2.2, 3.3.1, 3.4.2, 3.4.3 **SP800181**: T0414; K0003, K0039, K0044, K0157, K0168, K0177, K0211, K0260, K0261, K0262, K0524; S0010, S0357, S0368; A0033, A0123, A0151
	PO.1.2: Identify and document all security requirements for organization-developed software to meet, and maintain the requirements over time.	• Define policies that specify risk-based software architecture and design requirements, such as making code modular to facilitate code reuse and updates, isolating security components from other components during execution, avoiding undocumented commands and settings, and providing features that will aid software purchasers and consumers with the secure deployment, operation, and maintenance of the software. • Define policies that specify the security requirements for the organization's software, and verify compliance at key points in the SDLC (e.g., classes of software flaws verified by gates). • Analyze the risk of applicable technology stacks (e.g., languages, environments, deployment models), and recommend or require the use of stacks that will reduce risk compared to others. • Define policies that specify what needs to be archived for each software release (e.g., code, package files, third-party libraries, documentation) and how long it needs to be retained based on the SDLC model and other factors. • Ensure that 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. • Review all security requirements at least annually, or sooner if there are new requirements from internal or external sources or if a major vulnerability incident has occurred.	**BSAFSS**: SC.1-1, SC.2, PD.1-1, PD.1-2, PD.1-3, PD.2-2, SI, PA, CS, AA, LO, EE **BSIMM**: SM1.4, CP1.1, CP1.2, CP1.3 **IEC62443**: SR-3, SR-4, SR-5, SD-4 **ISO27034**: 7.3.2 **MSSDL**: 2, 5 **NISTCSF**: ID.GV-3 **OWASPMASVS**: 1.12 **OWASPSAMM**: PC1-A, PC1-B, PC2-A, PC3-A, SR1-A, SR1-B, SR2-B, SA1-B, IR1-A **PCISSLC**: 2.1, 2.2, 2.3, 3.3 **SCFPSSD**: Establish Coding Standards and Conventions **SP80053**: SA-2, SA-8 **SP800160**: 3.1.2, 3.2.1, 3.3.1 **SP800181**: T0414; K0003, K0039, K0044, K0157, K0168, K0177, K0211, K0260, K0261, K0262, K0524; S0010, S0357, S0368; A0033, A0123, A0151
	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]	• Define a core set of security requirements for software components, and include it in acquisition documents, software contracts, and other agreements with third parties. • Define security-related criteria for selecting software; the criteria can include things such as the third party's vulnerability disclosure program and product security incident response capabilities. • Require third parties to provide evidence that their software complies with the organization's security requirements. • Require third parties to provide provenance data for their software and its dependencies. • Establish and follow procedures to address risk when there are security requirements that third-party software components to be acquired do not meet.	**BSAFSS**: SM.1, SM.2, SM.2-1, SM.2-4 **BSIMM**: CP2.4, SR2.5, SR3.2 **IDASOAR**: 19**,** 21 **IEC62443**: SM-9, SM-10 **MSSDL**: 7 **NISTCSF**: ID.SC-3 **OWASPSAMM**: SR3-A **SCAGILE**: Tasks Requiring the Help of Security Experts 8 **SCFPSSD**: Manage Security Risk Inherent in the Use of Third-Party Components **SCSIC**: Vendor Sourcing Integrity Controls **SP80053:** SA-4, SA-9, SA-12, SR-5 **SP800160**: 3.1.1, 3.1.2 **SP800181**: T0203, T0415; K0039; S0374; A0056, A0161

Practices	Tasks	Implementation Examples	References
Implement Roles and Responsibilities (PO.2): Ensure that everyone inside and outside of the organization involved in the SDLC is prepared to perform their SSDF-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 SSDF. Periodically review and maintain the defined roles and responsibilities, updating them as needed.	• Define SSDF-related roles and responsibilities for all members of the software development team. • Integrate the security roles into the software development team. • Define roles and responsibilities for cybersecurity staff, security champions, project managers and leads, senior management, software developers, software testers, software assurance leads and staff, product owners, operations and platform engineers, and others involved in the SDLC. • Conduct an annual review of all roles and responsibilities. • Educate affected individuals on impending changes to roles and responsibilities.	**BSAFSS**: PD.2-1, PD.2-2 **BSIMM**: SM1.1, CP3.2 **IEC62443**: SM-2, SM-13 **NISTCSF**: ID.AM-6, ID.GV-2 **PCISSLC**: 1.2 **SCSIC**: Vendor Software Development Integrity Controls **SP80053**: SA-3 **SP800160**: 3.2.1, 3.2.4, 3.3.1 **SP800181**: K0233
	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.	• Document the desired outcomes of training for each role. • Define the type of training or curriculum required to achieve the desired outcome for each role. • Create a training plan for each role. • Acquire or create training for each role; acquired training may need to be customized for the organization. • Measure personnel performance to identify areas where changes to training may be beneficial.	**BSAFSS**: PD.2-2 **BSIMM**: SM1.3, CP2.5, T1.1, T1.5, T1.7, T1.8, T2.8, T3.2, T3.4 **IEC62443**: SM-4 **MSSDL**: 1 **NISTCSF**: PR.AT **OWASPSAMM**: EG1-A, EG2-A **PCISSLC**: 1.3 **SCAGILE:** Operational Security Tasks 14, 15; Tasks Requiring the Help of Security Experts 1 **SCFPSSD**: Planning the Implementation and Deployment of Secure Development Practices **SCSIC**: Vendor Software Development Integrity Controls **SP80053**: SA-8 **SP800160**: 3.2.4, 3.2.6 **SP800181**: OV-TEA-001, OV-TEA-002; T0030, T0073, T0320; K0204, K0208, K0220, K0226, K0243, K0245, K0252; S0100, S0101; A0004, A0057
	PO.2.3: Obtain upper management commitment to secure development, and convey that commitment to all with SSDF-related roles and responsibilities.	• Appoint a single leader or leadership team to be responsible for the entire secure software development process, including authorizing the release of software to production. • Increase upper management awareness of the risks of developing software without integrating security throughout the development life cycle and the risk mitigation provided by the SSDF practices. • Assist upper management in incorporating secure development support into their communications with personnel with SSDF-related roles and responsibilities. • Educate all personnel with SSDF-related roles and responsibilities on upper management's commitment to the SSDF and the importance of the SSDF to the organization.	**BSIMM**: SM1.2, SM1.3, CP2.5 **NISTCSF**: ID.RM-1, ID.SC-1 **OWASPSAMM**: SM1.A **PCISSLC**: 1.1 **SP800181**: T0001, T0004

Practices	Tasks	Implementation Examples	References
Implement Supporting Toolchains (PO.3): Use automation to reduce human effort and improve the accuracy, consistency, 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 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.3.1:** 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.	• Define categories of toolchains, and specify the mandatory tools or tool types to be used for each category. • Identify security tools to integrate into the developer toolchain. • Evaluate tools' signing capabilities to create immutable records/logs for auditability within the toolchain. • Use automated technology for toolchain management and orchestration.	**CNCFSSCP**: Securing Materials—Verification; Securing Build Pipelines—Verification, Automation, Secure Authentication/Access; Securing Artefacts—Verification; Securing Deployments—Verification **MSSDL**: 8 **OWASPSAMM**: IR2-B, ST2-B **SCAGILE**: Tasks Requiring the Help of Security Experts 9 **SCSIC**: Vendor Software Delivery Integrity Controls **SP80053**: SA-15 **SP800181**: K0013, K0178
	PO.3.2: Follow recommended security practices to deploy and maintain tools and toolchains.	• Evaluate, select, and acquire tools, and assess the security of each tool. • Integrate tools with other tools and existing software development processes and workflows. • Use code-based configuration for toolchains (e.g., pipelines as code, toolchains as code). • Implement the technologies and processes needed for reproducible builds. • Update, upgrade, or replace tools as needed to address tool vulnerabilities or add new tool capabilities. • Continuously monitor tools and tool logs for potential operational and security issues, including policy violations and anomalous behavior. • Regularly verify the integrity and check the provenance of each tool to identify potential problems. • Be prepared to share evidence and artifact data when requested with auditors and purchasers who want to confirm the use of tools and toolchains to support the development practices. • See PW.6 for examples of build and compilation tools.	**BSAFSS**: DE.2 **CNCFSSCP**: Securing Build Pipelines—Verification, Automation, Controlled Environments, Secure Authentication/Access; Securing Artefacts—Verification, Automation, Controlled Environments, Encryption; Securing Deployments—Verification, Automation **IEC62443**: SM-7 **NISTDVS**: 2.2 **OWASPASVS**: 1.14.3, 1.14.4, 14.1, 14.2 **OWASPMASVS**: 7.9 **OWASPSCVS**: 3, 5 **SCAGILE**: Tasks Requiring the Help of Security Experts 9 **SCFPSSD**: Use Current Compiler and Toolchain Versions and Secure Compiler Options **SCSIC**: Vendor Software Delivery Integrity Controls **SP80053**: SA-15 **SP800181**: K0013, K0178
	PO.3.3: Configure tools to generate evidence and artifacts of their support of secure software development practices as defined by the organization.	• Use existing tooling (e.g., workflow tracking, issue tracking, value stream mapping) to create an audit trail of the secure development-related actions that are performed for continuous improvement purposes. • Determine how often the collected information should be audited, and implement the necessary processes. • Establish and enforce security and retention policies for evidence and artifact data. • Be prepared to share evidence and artifact data when requested with auditors and purchasers who want to confirm the use of secure software development practices.	**BSAFSS**: PD.1-5 **CNCFSSCP**: Securing Build Pipelines—Verification, Automation, Controlled Environments; Securing Artefacts—Verification **IEC62443**: SM-12, SI-2 **MSSDL**: 8 **OWASPSAMM**: PC3-B **OWASPSCVS**: 3.13, 3.14 **PCISSLC**: 2.5 **SCAGILE**: Tasks Requiring the Help of Security Experts 9 **SCSIC**: Vendor Software Delivery Integrity Controls **SP80053**: SA-15 **SP800181**: K0013; T0024
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.	• Ensure that the criteria adequately indicate how effectively security risk is being managed. • Define key performance indicators (KPIs) and key risk indicators (KRIs) for software security. • 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 exception requests as part of the workflow and tracking system. • Summarize the results of the software security checks, including a description of the security risks that were successfully mitigated.	**BSAFSS**: TV.2-1, TV.5-1 **BSIMM**: SM1.4, SM2.2, SM2.6 **IEC62443**: SI-1, SI-2, SVV-3 **ISO27034**: 7.3.5 **MSSDL**: 3 **OWASPSAMM**: PC3-A, DR3-B, IR3-B, ST3-B **PCISSLC**: 3.3 **SP80053**: SA-15 **SP800160**: 3.2.1, 3.2.5, 3.3.1 **SP800181**: K0153, K0165

Practices	Tasks	Implementation Examples	References
	PO.4.2: Implement processes, mechanisms, etc. to gather and safeguard the necessary information in support of the criteria.	• 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. • Only allow authorized personnel to access the gathered information, and prevent any alteration or deletion of the information. • Be prepared to share evidence and artifact data when requested with auditors and purchasers who want to confirm the use of secure software development practices.	**BSAFSS**: PD.1-4, PD.1-5 **BSIMM**: SM1.4, SM2.2 **IEC62443**: SI-1, SVV-1, SVV-2, SVV-3, SVV-4 **OWASPSAMM**: PC3-B **PCISSLC**: 2.5 **SCSIC**: Vendor Software Delivery Integrity Controls **SP80053**: SA-15 **SP800160**: 3.2.5, 3.3.7 **SP800181**: T0349; K0153
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.	• Use separate identification and authentication realms with risk-based authentication and conditional access for each environment. • Use network segmentation and access controls to separate the environments from each other and from production environments, and to separate components from each other within each non-production environment, in order to reduce attack surfaces and attackers' lateral movement and privilege/access escalation. • Enforce authentication and tightly restrict connections entering and exiting each software development environment, including minimizing access to the internet to only what is necessary. • Minimize the use of and dependencies on production enterprise software in non-production environments. • Regularly log, monitor, and audit the trust relationships between the environments and between the components within each environment. • Continuously log and monitor operations and alerts across all components of the development environment to detect, respond, and recover from attempted and actual cyber incidents. • Configure security controls and other tools involved in separating and protecting the environments to generate evidence and artifacts for their activities. • Collect, protect, and regularly check provenance data for all software deployed in each environment, and determine if any of the software or their dependencies have new known vulnerabilities.	**BSAFSS**: DE.1, IA.1, IA.2 **CNCFSSCP**: Securing Build Pipelines—Controlled Environments **IEC62443**: SM-7 **NISTCSF:** PR.AC-5, PR.DS-7 **SCAGILE**: Tasks Requiring the Help of Security Experts 11 **SCSIC**: Vendor Software Delivery Integrity Controls **SP800181**: OM-NET-001, SP-SYS-001; T0019, T0023, T0144, T0160, T0262, T0438, T0484, T0485, T0553; K0001, K0005, K0007, K0033, K0049, K0056, K0061, K0071, K0104, K0112, K0179, K0326, K0487; S0007, S0084, S0121; A0048
	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.	• Configure each development endpoint based on approved hardening guides, checklists, etc.; for example, enable FIPS-compliant encryption of all sensitive data at rest and in transit. • Configure each development endpoint and the development resources to provide the least functionality needed by its users and services and to enforce the principle of least privilege. • Continuously monitor the security posture of all development endpoints. • Configure security controls and other tools involved in securing and hardening development endpoints to generate evidence and artifacts for their activities. • Require multi-factor authentication for all access to development endpoints and development resources. • Provide dedicated development endpoints on non-production networks for performing all development-related tasks; provide separate endpoints on production networks for typical enterprise tasks.	**BSAFSS:** DE.1-1, IA.1, IA.2 **IEC62443**: SM-7 **NISTCSF**: PR.AC-4, PR.AC-7, PR.IP-1, PR.IP-3, PR.IP-12, PR.PT-1, PR.PT-3, DE.CM **SCAGILE**: Tasks Requiring the Help of Security Experts 11 **SCSIC**: Vendor Software Delivery Integrity Controls **SP800181**: OM-ADM-001, SP-SYS-001; T0484, T0485, T0489, T0553; K0005, K0007, K0077, K0088, K0130, K0167, K0205, K0275; S0076, S0097, S0121, S0158; A0155

Practices	Tasks	Implementation Examples	References
Protect Software (PS)			
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 and executable code, based on the principle of least privilege so that only authorized personnel, tools, services, etc. have the necessary forms of access.	• 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 after the code has been automatically scanned for vulnerabilities and any issues have been remediated. • Use code signing to help protect the integrity of executables. • Use cryptography (e.g., cryptographic hashes) to help protect file integrity.	**BSAFSS**: IA.1, IA.2, SM.4-1, DE.1-2 **BSIMM**: SE2.4 **CNCFSSCP**: Securing the Source Code—Verification, Automation, Controlled Environments, Secure Authentication; Securing Materials—Automation **IDASOAR:** Fact Sheet 25 **IEC62443**: SM-6, SM-7, SM-8 **NISTCSF**: PR.AC-4, PR.DS-6, PR.IP-3 **OWASPASVS**: 1.10, 10.3.2 **OWASPMASVS**: 7.1 **OWASPSAMM**: OE3-B **PCISSLC**: 5.1, 6.1 **SCSIC**: Vendor Software Delivery Integrity Controls, Vendor Software Development Integrity Controls **SP80053**: SA-10
Provide a Mechanism for Verifying Software Release Integrity (PS.2): Help software purchasers and consumers ensure that the software they acquire is legitimate and has not been tampered with.	**PS.2.1:** Make integrity verification information available to software purchasers and consumers.	• Post cryptographic hashes for release files on a well-secured website. • Use an established certificate authority for code signing so that consumers' operating systems or other tools and services can confirm the validity of signatures before use. • Periodically review the code signing processes, including certificate renewal, rotation, revocation, and protection.	**BSAFSS**: SM.4, SM.5, SM.6 **BSIMM**: SE2.4 **CNCFSSCP**: Securing Deployments—Verification **IEC62443**: SM-6, SM-8, SUM-4 **NISTCSF**: PR.DS-6 **OWASPSAMM**: OE3-B **OWASPSCVS:** 4 **PCISSLC**: 6.1, 6.2 **SCSIC**: Vendor Software Delivery Integrity Controls **SP80053**: SR-9 **SP800181**: K0178
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 other data (e.g., integrity verification information, provenance data) to be retained for each software release.	• Store the release files, associated images, etc. in repositories following the organization's established policy; allow read-only access to them for auditing purposes by necessary personnel and no access by anyone else. • Store and protect release integrity verification information and provenance data, such as by keeping it in a separate location from the release files or by signing the data.	**BSAFSS**: PD.1-5, DE.1-2, IA.2 **CNCFSSCP**: Securing Artefacts—Automation, Controlled Environments, Encryption; Securing Deployments—Verification **IDASOAR:** 25 **IEC62443**: SM-6, SM-7 **NISTCSF**: PR.IP-4 **OWASPSCVS:** 1, 3.18, 3.19, 6.3 **PCISSLC**: 5.2, 6.1, 6.2 **SCSIC**: Vendor Software Delivery Integrity Controls **SP80053**: SA-10, SA-15, SR-9
	PS.3.2: Collect, maintain, and share provenance data for all components and other dependencies of each software release (e.g., in a software bill of materials [SBOM]).	• Make the provenance data available to software purchasers in accordance with your organization's policies, preferably using standards-based formats. • Update the provenance data every time any of the software's components or other dependencies are updated.	**BSAFSS**: SM.2 **BSIMM**: SE3.6 **CNCFSSCP**: Securing Materials—Verification, Automation **NTIASBOM**: All **OWASPSCVS**: 1.4, 2 **SCSIC**: Vendor Software Delivery Integrity Controls **SCTPC**: MAINTAIN3 **SP80053**: SR-4

Practices	Tasks	Implementation Examples	References
Produce Well-Secured Software (PW)			
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 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) helps make software development more efficient.	**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.	• Train the development team (security champions in particular) or collaborate with a risk modeling expert to create models and 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 to inform the security risk assessment. • Use data classification methods to identify and characterize each type of data that the software will interact with.	**BSAFSS**: SC.1 **BSIMM**: AM1.2, AM1.3, AM1.5, AM2.1, AM2.2, AM2.5, AM2.6, AM2.7 **IDASOAR**: 1 **IEC62443**: SM-4, SR-1, SR-2, SD-1 **ISO27034**: 7.3.3 **MSSDL**: 4 **NISTCSF**: ID.RA **NISTDVS**: 2.1 **OWASPASVS**: 1.1.2, 1.2, 1.4, 1.6, 1.8, 1.9, 1.11, 2, 3, 4, 6, 8, 9, 11, 12, 13 **OWASPMASVS**: 1.6, 1.8, 2, 3, 4, 5, 6 **OWASPSAMM**: TA1-A, TA1-B, TA3-B, DR1-A **PCISSLC**: 3.2, 3.3 **SCAGILE**: Tasks Requiring the Help of Security Experts 3 **SCFPSSD**: Threat Modeling **SCTTM**: Entire guide **SP80053**: SA-8, SA-15, SA-17 **SP800160**: 3.3.4, 3.4.5 **SP800181**: T0038, T0062; K0005, K0009, K0038, K0039, K0070, K0080, K0119, K0147, K0149, K0151, K0152, K0160, K0161, K0162, K0165, K0297, K0310, K0344, K0362, K0487, K0624; S0006, S0009, S0022, S0078, S0171, S0229, S0248; A0092, A0093, A0107
	PW.1.2: Document the software's security requirements, risks, and design decisions.	• Document the response to each risk, including how mitigations are to be achieved, and what the rationales are for any approved exceptions to the security requirements. • Summarize the documentation to serve as evidence and artifacts for the design activities.	**BSAFSS:** SC.1-1, PD.1-1 **BSIMM**: AA2.2 **IEC62443**: SD-1 **ISO27034**: 7.3.3 **MSSDL**: 4 **OWASPASVS**: 1.1.3, 1.1.4 **OWASPMASVS**: 1.3, 1.6 **OWASPSAMM**: DR1-B **PCISSLC**: 3.2, 3.3 **SP80053**: SA-10 **SP800181**: T0256; K0005, K0038, K0039, K0147, K0149, K0160, K0161, K0162, K0165, K0344, K0362, K0487; S0006, S0009, S0078, S0171, S0229, S0248; A0092, A0107
	PW.1.3: Where appropriate, build in support for using standardized security features and services (e.g., integrating 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]	• 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.	**BSAFSS**: SI.2-1, SI.2-2, LO.1 **BSIMM**: SFD1.1, SR1.1 **IEC62443**: SD-1, SD-4 **MSSDL**: 5 **OWASPASVS**: 1.1.6 **OWASPSAMM**: SA2-A **SCFPSSD**: Standardize Identity and Access Management; Establish Log Requirements and Audit Practices

Practices	Tasks	Implementation Examples	References
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 either 1) a qualified person (or people) who were not involved with the design or 2) adequate automated processes instantiated in the toolchain (or both) review the software design to confirm and enforce that it meets all of the security requirements and satisfactorily addresses the identified risk information.	• Review 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. • Document the findings of the design review to serve as evidence and artifacts.	**BSAFSS**: TV.3 **BSIMM**: AA1.1, AA1.2, AA2.1 **IEC62443**: SM-2, SR-2, SR-5, SD-3, SD-4, SI-2 **ISO27034**: 7.3.3 **OWASPASVS**: 1.1.5 **OWASPSAMM**: DR1-A, DR1-B **PCISSLC**: 3.2 **SP800181**: T0328; K0038, K0039, K0070, K0080, K0119, K0152, K0153, K0161, K0165, K0172, K0297; S0006, S0009, S0022, S0036, S0141, S0171
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.4.5*		
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 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.4.1:** Acquire 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.	• 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. • Obtain provenance information (e.g., SBOM, source composition analysis) for each software component, and analyze that information to better assess the risk that the component may introduce. • 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 along with their provenance data. • Designate which components must be included in software to be developed.	**BSAFSS**: SM.2 **BSIMM**: SR1.1 **CNCFSSCP**: Securing Materials—Verification **IDASOAR**: 19 **IEC62443**: SM-9, SM-10 **MSSDL**: 6 **NISTCSF**: ID.SC-2 **OWASPASVS**: 1.1.6 **OWASPSAMM**: SA1-A **OWASPSCVS**: 4 **SCSIC**: Vendor Sourcing Integrity Controls **SCTPC**: MAINTAIN **SP80053**: SA-4, SA-12 **SP800181**: K0039
	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.	• Follow organization-established security practices for secure software development when creating and maintaining the components. • Maintain an organization-wide software repository for these components. • Designate which components must be included in software to be developed.	**BSIMM**: SFD1.1, SFD2.1, SR1.1 **IDASOAR**: 19 **OWASPASVS**: 1.1.6 **SCTPC**: MAINTAIN **SP800181**: SP-DEV-001
	PW.4.3: *Moved to PW.1.3*		

Practices	Tasks	Implementation Examples	References
	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.	• Determine whether there are publicly known vulnerabilities in the software modules and services that vendors have not yet fixed. • Use existing results from commercial services for vetting the software modules and services. • [See PW.7 and PW.8]	**BSAFSS**: SC.3-1, TV.2, TV.3 **CNCFSSCP**: Securing Materials—Verification, Automation **IDASOAR**: 21 **IEC62443**: SI-1, DM-1 **MSSDL**: 7 **NISTCSF**: ID.SC-4 **NISTDVS**: 2.11 **OWASPASVS**: 10, 14.2 **OWASPMASVS**: 7.5 **OWASPSAMM**: TA3-A, SR3-B **OWASPSCVS:** 4, 5 **PCISSLC**: 3.2, 3.4, 4.1 **SCAGILE**: Tasks Requiring the Help of Security Experts 8 **SCFPSSD**: Manage Security Risk Inherent in the Use of Third-Party Components **SCSIC**: Vendor Sourcing Integrity Controls, Peer Reviews and Security Testing **SCTPC**: MAINTAIN, ASSESS **SP80053**: SA-9, SA-12, SR-3 **SP800160**: 3.1.2, 3.3.8 **SP800181**: SP-DEV-002; K0153, K0266
	PW.4.5: Verify the integrity and check the provenance of all in-house and third-party software components before reusing them for the organization's own software.	• Ensure each software component is still actively maintained and has not reached end of life; this should include new vulnerabilities found in the software being remediated. • Determine a plan of action for each software component that is no longer being maintained or will not be available in the near future. • Confirm the integrity of software components through digital signatures or other mechanisms.	**BSAFSS**: SM.2-1, SM.2-2, SM.2-3 **CNCFSSCP**: Securing Materials—Verification, Automation **IEC62443**: SM-9, SM-10 **NISTCSF**: PR.DS-6 **NISTDVS**: 2.11 **OWASPASVS**: 14.2.4, 14.2.5 **OWASPSCVS**: 4, 6 **SCSIC**: Vendor Sourcing Integrity Controls **SP80053**: SR-4 **SP800181**: S0298
Create Source Code by Adhering to Secure Coding Practices (PW.5): Decrease the number of security vulnerabilities in the software, and reduce costs by eliminating vulnerabilities during source code creation by following 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.	• 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 automated features that encourage or require the use of secure coding practices with just-in-time training-in-place. • Follow procedures for manually ensuring compliance with secure coding practices. • Check for other vulnerabilities that are common to the development languages and environment. • Have the developer review their own human-readable code to complement (not replace) code review performed by other people or tools. [See PW.7]	**BSAFSS**: SC.2, SC.3, LO.1, EE.1 **BSIMM**: SR3.3, CR3.5 **IDASOAR**: 2 **IEC62443**: SI-1, SI-2 **ISO27034**: 7.3.5 **MSSDL**: 9 **OWASPASVS**: 1.1.7, 1.5, 1.7, 5, 7 **OWASPMASVS**: 7.6 **SCFPSSD**: Establish Log Requirements and Audit Practices, Use Code Analysis Tools to Find Security Issues Early, Handle Data Safely, Handle Errors, Use Safe Functions Only **SP800181**: SP-DEV-001; T0013, T0077, T0176; K0009, K0016, K0039, K0070, K0140, K0624; S0019, S0060, S0149, S0172, S0266**;** A0036, A0047
	PW.5.2: *Moved to PW.5.1 as example*		

Practices	Tasks	Implementation Examples	References
Configure the Integrated Development Environment, 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.	• Use up-to-date versions of compiler, interpreter, and build tools. • Follow change management processes when deploying or updating compiler, interpreter, and build tools, and audit all unexpected changes to tools. • Regularly validate the authenticity and integrity of compiler, interpreter, and build tools.	**BSAFSS**: DE.2-1 **CNCFSSCP**: Securing Build Pipelines—Verification, Automation **IEC62443**: SI-2 **MSSDL**: 8 **SCAGILE**: Operational Security Task 3 **SCFPSSD**: Use Current Compiler and Toolchain Versions and Secure Compiler Options **SCSIC**: Vendor Software Development Integrity Controls **SP80053**: 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.	• 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. • Continuously verify that the approved configurations are being used. • Document information about the compiler, interpreter, and build tool configuration in a knowledge base that developers can access, search, and reproduce in their local development environment.	**BSAFSS**: DE.2-3, DE.2-4, DE.2-5 **CNCFSSCP**: Securing Build Pipelines—Verification, Automation **IEC62443**: SI-2 **MSSDL**: 8 **NISTDVS**: 2.5 **OWASPASVS**: 14.1, 14.2.1 **OWASPMASVS**: 7.2 **PCISSLC**: 3.2 **SCAGILE**: Operational Security Task 8 **SCFPSSD**: Use Current Compiler and Toolchain Versions and Secure Compiler Options **SCSIC**: Vendor Software Development Integrity Controls **SP80053**: SA-15 **SP800181**: K0039, K0070
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-readable.	**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.	• Follow the organization’s policies or guidelines for when code review should be performed and how it should be conducted. This may include 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.	**IEC62443**: SM-5, SI-1, SVV-1 **SCSIC**: Peer Reviews and Security Testing **SP80053**: SA-11 **SP800181**: SP-DEV-002; K0013, K0039, K0070, K0153, K0165; S0174
	PW.7.2: 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.	• 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 compliance with the organization’s secure coding standards with a human reviewing the 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.	**BSAFSS**: TV.2, PD.1-4 **BSIMM**: CR1.2, CR1.4, CR1.6, CR2.6, CR2.7, CR3.5 **IDASOAR**: 3, 4, 5, 14, 15, 48 **IEC62443**: SI-1, SVV-1, SVV-2 **ISO27034**: 7.3.6 **MSSDL**: 9, 10 **NISTDVS**: 2.3, 2.4 **OWASPASVS**: 1.1.7, 10 **OWASPMASVS**: 7.5 **OWASPSAMM**: IR1-B, IR2-A, IR2-B, IR3-A **PCISSLC**: 3.2, 4.1 **SCAGILE**: Operational Security Tasks 4, 7; Tasks Requiring the Help of Security Experts 10 **SCFPSSD**: Use Code Analysis Tools to Find Security Issues Early, Use Static Analysis Security Testing Tools, Perform Manual Verification of Security Features/Mitigations **SCSIC**: Peer Reviews and Security Testing **SP80053**: SA-11, SA-15 **SP800181**: SP-DEV-001, SP-DEV-002; T0013, T0111, T0176, T0267, T0516; K0009, K0039, K0070, K0140, K0624; S0019, S0060, S0078, S0137, S0149, S0167, S0174, S0242, S0266**;** A0007, A0015, A0036, A0044, A0047

Practices	Tasks	Implementation Examples	References
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. 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 if executable code testing should be performed to identify and eliminate classes of vulnerabilities not covered by previous reviews, analysis, or testing, and if so, which types should be used.	• Follow the organization's policies or guidelines for when code testing should be performed and how it should be conducted (e.g., within a sandboxed environment). This may include third-party executable code and reusable executable code modules written in-house.	**BSAFSS**: TV.3 **IEC62443**: SVV-1, SVV-2, SVV-3, SVV-4, SVV-5 **SCSIC**: Peer Reviews and Security Testing **SP80053**: SA-11 **SP800181**: SP-DEV-001, SP-DEV-002; T0456; K0013, K0039, K0070, K0153, K0165, K0342, K0367, K0536, K0624; S0001, S0015, S0026, S0061, S0083, S0112, S0135
	PW.8.2: Design the tests, perform the testing, and document the results, including documenting and triaging all discovered issues and recommended remediations in the development team's workflow or issue tracking system.	• 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 test suite to ensure that errors are not reintroduced. • Take into consideration the infrastructures and technology stacks that the software will be used with in production when developing test plans. • Use 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.	**BSAFSS**: TV.3, TV.5, PD.1-4 **BSIMM**: ST1.1, ST1.3, ST2.1, ST2.4, ST2.5, ST2.6, ST3.3, ST3.4, ST3.5, ST3.6, PT1.1, PT1.2, PT1.3 **IDASOAR**: 7, 8, 10, 11, 38, 39, 43, 44, 48, 55, 56, 57 **IEC62443**: SM-5, SM-13, SI-1, SVV-1, SVV-2, SVV-3, SVV-4, SVV-5 **ISO27034**: 7.3.6 **MSSDL**: 10, 11 **NISTDVS**: 2.6, 2.7, 2.8, 2.9, 2.10, 2.11 **OWASPMASVS**: 7.5 **OWASPSAMM**: ST1-A, ST1-B, ST2-A, ST2-B, ST3-A **PCISSLC**: 4.1 **SCAGILE**: Operational Security Tasks 10, 11; Tasks Requiring the Help of Security Experts 4, 5, 6, 7 **SCFPSSD**: Perform Dynamic Analysis Security Testing, Fuzz Parsers, Network Vulnerability Scanning, Perform Automated Functional Testing of Security Features/Mitigations, Perform Penetration Testing **SCSIC**: Peer Reviews and Security Testing **SP80053:** SA-11, SA-15 **SP800181**: SP-DEV-001, SP-DEV-002; T0013, T0028, T0169, T0176, T0253, T0266, T0456, T0516; K0009, K0039, K0070, K0272, K0339, K0342, K0362, K0536, K0624; S0001, S0015, S0046, S0051, S0078, S0081, S0083, S0135, S0137, S0167, S0242; A0015
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 so that the default settings are secure and do not weaken the security functions provided by the platform, network infrastructure, or services.	• 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.	**BSAFSS**: CF.1 **IDASOAR**: 23 **IEC62443**: SD-4, SVV-1, SG-1 **ISO27034**: 7.3.5 **SCAGILE**: Tasks Requiring the Help of Security Experts 12 **SCSIC**: Vendor Software Delivery Integrity Controls, Vendor Software Development Integrity Controls **SP800181**: SP-DEV-002; K0009, K0039, K0073, K0153, K0165, K0275, K0531; S0167
	PW.9.2: Implement the default settings (or groups of default settings, if applicable), and document each setting for software administrators.	• 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. • Use authoritative programmatic technical mechanisms to document how each setting can be implemented and assessed by software administrators. • Store the default configuration in a usable format and follow change control practices for modifying it (e.g., configuration as code).	**BSAFSS**: CF.1 **BSIMM**: SE2.2 **IDASOAR**: 23 **IEC62443**: SG-3 **OWASPSAMM**: OE1-A **PCISSLC**: 8.1, 8.2 **SCAGILE**: Tasks Requiring the Help of Security Experts 12 **SCFPSSD**: Verify Secure Configurations and Use of Platform Mitigation **SCSIC**: Vendor Software Delivery Integrity Controls, Vendor Software Development Integrity Controls **SP800181**: SP-DEV-001; K0009, K0039, K0073, K0153, K0165, K0275, K0531

Practices	Tasks	Implementation Examples	References
Respond to Vulnerabilities (RV)			
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, reducing the window of opportunity for attackers.	**RV.1.1:** Gather information from purchasers, consumers, and public sources on potential vulnerabilities in the software and third-party components that the software uses, and investigate all credible reports.	• Establish a vulnerability disclosure program, and make it easy for security researchers to learn about your program and report possible vulnerabilities. • 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. • Regularly check the provenance and software composition data for each software release in use to identify potential new vulnerabilities in its components.	**BSAFSS**: VM.1-3, VM.3 **BSIMM**: CMVM1.2, CMVM3.4 **CNCFSSCP**: Securing Materials—Verification **IEC62443**: DM-1, DM-2, DM-3 **ISO29147**: 6.2.1, 6.2.2, 6.2.4, 6.3, 6.5 **ISO30111**: 7.1.3 **OWASPSAMM**: IM1-A, IM2-B, EH1-B **OWASPSCVS:** 4 **PCISSLC**: 3.4, 4.1, 9.1 **SCAGILE**: Operational Security Task 5 **SCFPSSD**: Vulnerability Response and Disclosure **SCTPC**: MONITOR1 **SP800181**: K0009, K0038, K0040, K0070, K0161, K0362; S0078
	RV.1.2: Review, analyze, and/or test the software's code to identify or confirm the presence of previously undetected vulnerabilities.	• Configure the toolchain to perform automated code analysis and testing on a regular or continuous basis. • Automatically review provenance and software composition data for all software components and dependencies to identify any new vulnerabilities they have. • [See PW.7 and PW.8]	**BSAFSS**: VM.1-2, VM.2-1 **IEC62443**: SI-1, SVV-2, SVV-3, SVV-4, DM-1, DM-2 **ISO27034**: 7.3.6 **ISO29147**: 6.4 **ISO30111**: 7.1.4 **PCISSLC**: 3.4, 4.1 **SCAGILE**: Operational Security Tasks 10, 11 **SP80053**: SA-11 **SP800181**: SP-DEV-002; K0009, K0039, K0153
	RV.1.3: Have a policy that addresses vulnerability disclosure and remediation, and implement the roles, responsibilities, and processes needed to support that policy.	• Have a Product Security Incident Response Team (PSIRT) and processes in place to handle the responses to vulnerability reports and incidents. • 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 and open-source software components.	**BSAFSS**: VM.1-1, VM.2 **BSIMM**: CMVM1.1, CMVM2.1 **IEC62443**: DM-1, DM-2, DM-3, DM-4, DM-5 **ISO29147**: All **ISO30111**: All **MSSDL**: 12 **OWASPMASVS**: 1.11 **OWASPSAMM**: IM1-A, IM1-B, IM2-A, IM2-B **PCISSLC**: 9.2, 9.3 **SCFPSSD**: Vulnerability Response and Disclosure **SP800160**: 3.3.8 **SP800181**: K0041, K0042, K0151, K0292, K0317; S0054; A0025
Assess, Prioritize, and Remediate Vulnerabilities (RV.2): Help ensure that vulnerabilities are remediated as quickly as necessary, reducing the window of opportunity for attackers.	**RV.2.1:** Analyze each vulnerability to gather sufficient information to plan its remediation.	• Use issue tracking software (existing software, if available) to document each vulnerability. • Estimate how much effort would be required to remediate the vulnerability. • Estimate the potential impact of vulnerability exploitation. • Estimate the resources needed to weaponize the vulnerability if that has not already been done. • Estimate any other relevant factors needed to plan the remediation of the vulnerability.	**BSAFSS**: VM.2 **BSIMM**: CMVM1.2, CMVM2.2 **IEC62443**: DM-2, DM-3 **ISO30111**: 7.1.4 **PCISSLC**: 3.4, 4.2 **SCAGILE**: Operational Security Task 1, Tasks Requiring the Help of Security Experts 10 **SP80053**: SA-10 **SP800160**: 3.3.8 **SP800181**: K0009, K0039, K0070, K0161, K0165; S0078

Practices	Tasks	Implementation Examples	References
	RV.2.2: Develop and implement a remediation plan for each vulnerability.	• Make a risk-based decision as to whether the vulnerability will be remediated or if the risk will be addressed through other means (e.g., risk acceptance, risk transference), and prioritize any actions to be taken • 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. • Develop and release security advisories that provide the necessary information to software purchasers and consumers, including descriptions of what has changed in the software and what configuration settings might need to be changed, if any. • Deliver the remediation to the purchasers and consumers via an automated and trusted delivery mechanism.	**BSAFSS**: VM.1-1, VM-2 **IEC62443**: DM-4 **ISO30111**: 7.1.4, 7.1.5 **PCISSLC**: 4.1, 4.2, 10.1 **SCAGILE**: Operational Security Task 2 **SCFPSSD**: Fix the Vulnerability, Identify Mitigating Factors or Workarounds **SCTPC**: MITIGATE **SP800160**: 3.3.8 **SP800181**: T0163, T0229, T0264; K0009, K0070
Analyze Vulnerabilities to Identify Their Root Causes (RV.3): Help reduce the frequency of vulnerabilities in the future.	**RV.3.1:** Analyze all identified vulnerabilities to determine the root cause of each vulnerability.	• Document the root cause of each discovered issue. • Document lessons learned through root cause analysis in a knowledge base that developers can access and search.	**BSAFSS**: VM.2-1 **IEC62443**: DM-3 **ISO30111**: 7.1.4 **OWASPSAMM**: IM3-A **PCISSLC**: 4.2 **SCFPSSD**: Secure Development Lifecycle Feedback **SP800181**: T0047, K0009, K0039, K0070, K0343
	RV.3.2: Analyze the root causes over time to identify patterns, such as a particular secure coding practice not being followed consistently.	• Document lessons learned through 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.	**BSAFSS**: VM.2-1, PD.1-3 **IEC62443**: DM-4 **ISO30111**: 7.1.7 **OWASPSAMM**: IM3-B **PCISSLC**: 2.6, 4.2 **SCFPSSD**: Secure Development Lifecycle Feedback **SP80053**: SA-15 **SP800160**: 3.3.8 **SP800181**: T0111, K0009, K0039, K0070, K0343
	RV.3.3: Review the software for similar vulnerabilities, and proactively fix them rather than waiting for external reports.	• [See PW.7 and PW.8]	**BSAFSS**: VM.2 **BSIMM**: CMVM3.1 **IEC62443**: SI-1, DM-3, DM-4 **ISO30111**: 7.1.4 **PCISSLC**: 4.2 **SP800181**: SP-DEV-001, SP-DEV-002; K0009, K0039, K0070
	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.	• Document lessons learned through root cause analysis in a knowledge base that developers can access and search. • Plan and implement changes to the appropriate SSDF practices.	**BSAFSS**: PD.1-3 **BSIMM**: CMVM3.2 **IEC62443**: DM-6 **ISO30111**: 7.1.7 **MSSDL**: 2 **PCISSLC**: 2.6, 4.2 **SCFPSSD**: Secure Development Lifecycle Feedback **SP80053**: SA-15 **SP800181**: K0009, K0039, K0070

References

[BSAFSS] BSA (2020) *The BSA Framework for Secure Software: A New Approach to Securing the Software Lifecycle, Version 1.1.* Available at https://www.bsa.org/files/reports/bsa_framework_secure_software_update_2020.pdf

[BSIMM] Migues S, Steven J, Ware M (2020) *Building Security in Maturity Model (BSIMM) Version 11*. Available at https://www.bsimm.com/download/

[CNCFSSCP] Cloud Native Computing Foundation (2021) *Software Supply Chain Best Practices.* Available at https://github.com/cncf/tag-security/tree/main/supply-chain-security/supply-chain-security-paper

[IDASOAR] Hong Fong EK, Wheeler D, Henninger A (2016) State-of-the-Art Resources (SOAR) for Software Vulnerability Detection, Test, and Evaluation 2016. (Institute for Defense Analyses [IDA], Alexandria, VA), IDA Paper P-8005. Available at https://www.ida.org/research-and-publications/publications/all/s/st/stateoftheart-resources-soar-for-software-vulnerability-detection-test-and-evaluation-2016

[IEC62443] International Electrotechnical Commission (IEC), Security for industrial automation and control systems – Part 4-1: Secure product development lifecycle requirements, IEC 62443-4-1, 2018. Available at https://webstore.iec.ch/publication/33615

[ISO27034] International Organization for Standardization (ISO)/IEC, Information technology – Security techniques – Application security – Part 1: Overview and concepts, ISO/IEC 27034-1:2011, 2011. Available at https://www.iso.org/standard/44378.html

[ISO29147] International Organization for Standardization (ISO)/IEC, Information technology – Security techniques – Vulnerability disclosure, ISO/IEC 29147:2018, 2018. Available at https://www.iso.org/standard/72311.html

[ISO30111] International Organization for Standardization (ISO)/IEC, Information technology – Security techniques – Vulnerability handling processes, ISO/IEC 30111:2019, 2019. Available at https://www.iso.org/standard/69725.html

[MSSDL] Microsoft (2021) *Security Development Lifecycle*. Available at https://www.microsoft.com/en-us/securityengineering/sdl/

[NISTCSF] National Institute of Standards and Technology (2018) Framework for Improving Critical Infrastructure Cybersecurity, Version 1.1. (National Institute of Standards and Technology, Gaithersburg, MD). https://doi.org/10.6028/NIST.CSWP.04162018

[NISTDVS] Black P, Guttman B, Okun V (2021) Guidelines on Minimum Standards for Developer Verification of Software. (National Institute of Standards and Technology, Gaithersburg, MD). Available at https://www.nist.gov/system/files/documents/2021/07/13/Developer%20Verification%20of%20Software.pdf

[NTIASBOM] National Telecommunications and Information Administration (NTIA) (2021) *The Minimum Elements For a Software Bill of Materials (SBOM)*. Available at https://www.ntia.doc.gov/report/2021/minimum-elements-software-bill-materials-sbom

[OWASPASVS] Open Web Application Security Project (2020) *OWASP Application Security Verification Standard 4.0.2*. Available at https://github.com/OWASP/ASVS

[OWASPMASVS] Open Web Application Security Project (2021) *OWASP Mobile Application Security Verification Standard, Version 1.3*. Available at https://github.com/OWASP/owasp-masvs/releases

[OWASPSAMM] Open Web Application Security Project (2017) *Software Assurance Maturity Model Version 1.5*. Available at https://www.owasp.org/index.php/OWASP_SAMM_Project

[OWASPSCVS] Open Web Application Security Project (2020) *OWASP Software Component Verification Standard, Version 1.0*. Available at https://github.com/OWASP/Software-Component-Verification-Standard

[PCISSLC] Payment Card Industry (PCI) Security Standards Council (2021) *Secure Software Lifecycle (Secure SLC) Requirements and Assessment Procedures Version 1.1*. Available at https://www.pcisecuritystandards.org/document_library?category=sware_sec#results

[SCAGILE] Software Assurance Forum for Excellence in Code (2012) *Practical Security Stories and Security Tasks for Agile Development Environments*. Available at http://www.safecode.org/publication/SAFECode_Agile_Dev_Security0712.pdf

[SCFPSSD] Software Assurance Forum for Excellence in Code (2018) *Fundamental Practices for Secure Software Development: Essential Elements of a*

Secure Development Lifecycle Program, Third Edition. Available at https://safecode.org/wp-content/uploads/2018/03/SAFECode_Fundamental_Practices_for_Secure_Software_Development_March_2018.pdf

[SCSIC] Software Assurance Forum for Excellence in Code (2010) *Software Integrity Controls: An Assurance-Based Approach to Minimizing Risks in the Software Supply Chain*. Available at http://www.safecode.org/publication/SAFECode_Software_Integrity_Controls0610.pdf

[SCTPC] Software Assurance Forum for Excellence in Code (2017) *Managing Security Risks Inherent in the Use of Third-Party Components*. Available at https://www.safecode.org/wp-content/uploads/2017/05/SAFECode_TPC_Whitepaper.pdf

[SCTTM] Software Assurance Forum for Excellence in Code (2017) *Tactical Threat Modeling*. Available at https://www.safecode.org/wp-content/uploads/2017/05/SAFECode_TM_Whitepaper.pdf

[SP80053] Joint Task Force (2020) Security and Privacy Controls for Information Systems and Organizations. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-53, Rev. 5. Includes updates as of December 10, 2020. https://doi.org/10.6028/NIST.SP.800-53r5

[SP800160] Ross R, McEvilley M, Oren J (2016) Systems Security Engineering: Considerations for a Multidisciplinary Approach in the Engineering of Trustworthy Secure Systems. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-160, Volume 1, Includes updates as of March 21, 2018. https://doi.org/10.6028/NIST.SP.800-160v1

[SP800181] Newhouse W, Keith S, Scribner B, Witte G (2017) National Initiative for Cybersecurity Education (NICE) Cybersecurity Workforce Framework. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-181. https://doi.org/10.6028/NIST.SP.800-181

Appendix A—The SSDF and Executive Order 14028

The President's Executive Order (EO) on "Improving the Nation's Cybersecurity (14028)" issued on May 12, 2021, charged multiple agencies – including NIST – with enhancing cybersecurity through a variety of initiatives related to the security and integrity of the software supply chain.

Section 4 of the EO directed NIST to solicit input from the private sector, academia, government agencies, and others and to identify existing or develop new standards, tools, best practices, and other guidelines to enhance software supply chain security. Table 2 maps the clauses from Section 4 of the EO to the SSDF practices and tasks that help address each clause.

Table 2: SSDF Practices Corresponding to EO 14028 Clauses

EO 14028 Clause	SSDF Practices and Tasks
4(c)	All practices and tasks
4(e)(i)(A)	PO.5.1
4(e)(i)(B)	PO.5.1
4(e)(i)(C)	PO.5.2
4(e)(i)(D)	PO.5.1
4(e)(i)(E)	PO.5.2
4(e)(i)(F)	PO.3.2, PO.3.3, PO.5.1, PO.5.2
4(e)(ii)	PO.3.2, PO.3.3, PO.5.1, PO.5.2
4(e)(iii)	PO.3.1, PO.3.2, PO.5.1, PO.5.2, PS.1.1, PS.2.1, PS.3.1, PW.4.5
4(e)(iv)	PO.4.1, PO.4.2, PS.1.1, PW.2.1, PW.4.4, PW.5.1, PW.6.1, PW.6.2, PW.7.1, PW.7.2, PW.8.2, PW.9.1, PW.9.2, RV.1.1, RV.1.2, RV.2.1, RV.2.2, RV.3.3
4(e)(v)	PO.3.2, PO.3.3, PO.4.1, PO.5.1, PO.5.2, PW.1.2, PW.2.1, PW.7.2, PW.8.2, RV.2.2
4(e)(vi)	PO.1.3, PO.3.2, PO.5.1, PS.3.1, PS.3.2, PW.4.1, PW.4.5, RV.1.2
4(e)(vii)	PS.3.2
4(e)(viii)	RV.1.1, RV.1.2, RV.1.3, RV.2.1, RV.2.2, RV.3.3
4(e)(ix)	All practices and tasks
4(e)(x)	PO.1.3, PS.3.2, PW.4.1, PW.4.4, PW.4.5

Appendix B—Acronyms

Selected acronyms and abbreviations used in this document are defined below.

BSIMM	Building Security In Maturity Model
CISQ	Consortium for Information & Software Quality
CNCF	Cloud Native Computing Foundation
COTS	Commercial-Off-the-Shelf
CPS	Cyber-Physical System
DevOps	Development and Operations
EO	Executive Order
GOTS	Government-Off-the-Shelf
ICS	Industrial Control System
IDA	Institute for Defense Analyses
IEC	International Electrotechnical Commission
IoT	Internet of Things
ISO	International Organization for Standardization
ISPAB	Information Security and Privacy Advisory Board
IT	Information Technology
ITL	Information Technology Laboratory
KPI	Key Performance Indicator
KRI	Key Risk Indicator
MITA	Medical Imaging & Technology Alliance
NAVSEA	Naval Sea Systems Command
NICE	National Initiative for Cybersecurity Education
NIST	National Institute of Standards and Technology
NTIA	National Telecommunications and Information Administration
OWASP	Open Web Application Security Project
PCI	Payment Card Industry
PSIRT	Product Security Incident Response Team
SAFECode	Software Assurance Forum for Excellence in Code
SAMM	Software Assurance Maturity Model
SBOM	Software Bill of Materials
SDL	[Microsoft] Security Development Lifecycle
SDLC	Software Development Life Cycle
SEI	Software Engineering Institute
SLC	Software Lifecycle
SOAR	State-of-the-Art Resources
SSDF	Secure Software Development Framework

Appendix C—Change Log

This appendix summarizes the most noteworthy changes from the original SSDF, published in April 2020, to this draft:

- References
  - Added CNCFSSCP, IEC62443, ISO29147, ISO30111, NISTDVS, OWASPMASVS, OWASPSCVS
  - Updated BSAFSS, BSIMM, OWASPASVS, PCISSLC
  - Deleted OWASPTEST
- Practices
  - Added PO.5
  - Deleted PW.3 (merged into PW.4)
- Tasks
  - Added PO.1.2, PO.5.1, PO.5.2, PS.3.2, PW.1.2
  - Moved PW.3.1 to PO.1.3; moved PW.3.2 to PW.4.5; moved PW.4.3 to PW.1.3
  - Demoted PW.5.2 to a PW.5.1 example
- SSDF Table Conventions
  - Retired identifiers for deleted/moved practices and tasks (PW.3, PW.3.1, PW.3.2, PW.4.3, and PW.5.2)
  - Added colored borders and shaded rows for each group of practices; indicated retired practices and tasks by a lack of shading
- Converted the content from a white paper to a Special Publication 800-series document
- Added Appendix A