PROBLEM STATEMENT 2

Approach for reducing vulnerabilities in products, software and Applications.

KEY FEATURES

- Secure Software Development Framework (SSDF)
- Advantages of high level SSDF practices
- White Paper
- White Paper Practices
- Elements of each Practice
- Practices Tasks and Implementations

SSDF

The Secure Software Development Framework (SSDF) is a set of practices that organizations can adopt to integrate security into their software development lifecycle (SDLC). It provides guidance on how to develop secure software, reduce vulnerabilities, and manage risks throughout the development process. The SSDF is particularly relevant as the software supply chain becomes a critical attack surface, and organizations seek to ensure that their software is free of known vulnerabilities and security flaws.

White Paper

This white paper introduces a software development framework (SSDF) of fundamental, sound, and secure software development practices based on established secure software development practice documents

White Paper Practices

- Prepare the Organisation
- Protect the Software
- Produce Well Secured Software
- Respond to Vulnerabilities

- **Prepare the Organization (PO)**: Ensure that the organization’s people, processes, and technology are prepared to perform secure software development at the organization level and, in some cases, for each individual project.
- **Protect the Software (PS)**: Protect all components of the software from tampering and unauthorized access.
- **Produce Well-Secured Software (PW)**: Produce well-secured software that has minimal security vulnerabilities in its releases.
- **Respond to Vulnerabilities (RV)**: Identify vulnerabilities in software releases and respond appropriately to address those vulnerabilities and prevent similar vulnerabilities from occurring in the future.

ELEMENTS OF EACH PRACTICE

Practice

A brief statement of the practice, along with a unique identifier and an explanation of what the practice is and why it is beneficial.

TASK

An individual action (or actions) needed to accomplish a practice.

Implementation Example

An example of a type of tool, process, or other method that could be used to implement this practice; not intended to imply that any example or combination of examples is required or that only the stated examples are feasible options.

Reference

An established secure development practice document and its mappings to a particular task.

Practice	Tasks
Prepare the Organisation	
Define Security Requirements for Software Development	Identify all applicable security requirements for the organization’s general software development, and maintain the requirements over time.
Implement Roles and Responsibilities	• Create new roles and alter responsibilities for existing roles to encompass all parts of the SSDF. Periodically review the defined roles and responsibilities, and update them as needed. • Provide role-specific training for all personnel with responsibilities that contribute to secure development. Periodically review role-specific training and update it as needed.

Implement a Supporting Toolchain	• Specify which tools or tool types are to be included in each toolchain and which are mandatory, as well as how the toolchain components are to be integrated with each other. • Following sound security practices, deploy and configure tools, integrate them within the toolchain, and maintain the individual tools and the toolchain as a whole. • Configure tools to collect evidence and artifacts of their support of the secure software development practices
Define Criteria for Software Security Checks	• Define criteria for software security checks throughout the SDLC. • Implement processes, mechanisms, etc. to gather the necessary information in support of the criteria.

Practice	Tasks
Protect Software	
Protect All Forms of Code from Unauthorized Access and Tampering	Store all forms of code, including source code and executable code, based on the principle of least privilege so that only authorized personnel have the necessary forms of access.
Provide a Mechanism for Verifying Software Release Integrity	Make verification information available to software consumers.
Archive and Protect Each Software Release	Securely archive a copy of each release and all of its components (e.g., code, package files, third-party libraries, documentation), and release integrity verification information.

Practice	Tasks
Produce Well-Secured Software (PW)	
Design Software to Meet Security Requirements and Mitigate Security Risks	Use forms of risk modeling, such as threat modeling, attack modeling, or attack surface mapping, to help assess the security risk for the software.
Review the Software Design to Verify Compliance with Security Requirements and Risk Information	Have a qualified person who was not involved with the software design review it to confirm that it meets all of the security requirements and satisfactorily addresses the identified risk information
Verify Third-Party Software Complies with Security Requirements	Communicate requirements to third parties who may provide software modules and services to the organization for reuse by the organization's own software.

Reuse Existing, Well-Secured Software When Feasible Instead of Duplicating Functionality	Acquire well-secured components (e.g., software libraries, modules, middleware, frameworks) from third parties for use by the organization’s software.
Create Source Code Adhering to Secure Coding Practices criteria for Software Security Checks	• Follow all secure coding practices that are appropriate to the development languages and environment. • Have the developer review their own humanreadable code, analyze their own human-readable code, and/or test their own executable code to complement (not replace) code review, analysis, and/or testing performed by others.

Configure the Compilation and Build Processes to Improve Executable Security

- Use compiler and build tools that offer features to improve executable security.
- Determine which compiler and build tool features should be used and how each should be configured, then implement the approved configuration for compilation and build tools, processes, etc.

Review and/or Analyze Human-Readable Code to Identify Vulnerabilities and Verify Compliance with Security Requirements

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.

Test Executable Code to Identify Vulnerabilities and Verify Compliance with Security Requirements

- Determine if executable code testing should be performed and, if so, which types should be used.
- Design the tests, perform the testing, and document the results

Configure the Software to Have Secure Settings by Default

Determine 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.

Practice	Tasks
Respond to Vulnerabilities (RV)	
Identify and Confirm Vulnerabilities on an Ongoing Basis	• Gather information from consumers and public sources on potential vulnerabilities in the software and any third-party components that the software uses, and investigate all credible reports. • Review, analyze, and/or test the software’s code to identify or confirm the presence of previously undetected vulnerabilities. • Have a team and process in place to handle the responses to vulnerability reports and incidents.

Assess, Prioritize, and Remediate Vulnerabilities	• Analyze each vulnerability to gather sufficient information to plan its remediation. • Develop and implement a remediation plan for each vulnerability.
Analyze Vulnerabilities to Identify Their Root Causes	• Analyze all identified vulnerabilities to determine the root cause of each vulnerability. • Analyze the root causes over time to identify patterns, such as when a particular secure coding practice is not being followed consistently. • Review the SDLC process, and update it as appropriate to prevent (or reduce the likelihood of) the root cause recurring in updates to this software or in new software that is created.

THANK YOU