NIST SSDF Distilled

Martin (Marty) VanWinkle

- System / Security Administrator @ ias.edu
- Has been known to program things.
  - 2000-2005 - Arizona Research Laboratories (arl.arizona.edu)
  - 2006-2013 - Corporate
  - 2013-Present - ias.edu
- All materials for this talk can be found here:
  - people.ias.edu/~mvanwinkle
  - github.com/mvanwinkleias/talks

Jay Gallman

- Risk Advisor @ duke.edu
  - Can we comply with?
  - Familiarity with NIST especially SP 800-171
  - Vendor risk assessments - Do you know what third party components lurk in that platform?
- Comfortable at 40000, 20000, and every so often on the ground itself

About NIST

- The National Institute of Standards and Technology (NIST) was founded in 1901 and is now part of the U.S. Department of Commerce. NIST is one of the nation's oldest physical science laboratories.
- Mission: To promote U.S. innovation and industrial competitiveness by advancing measurement science, standards, and technology in ways that enhance economic security and improve our quality of life.

(obtained from https://www.nist.gov/about-nist)

Overall Goals - High Level

- Reduce the barrier to entry to the SSDF
  - Cognitive load
  - Accessibility
- The audience should
  - have a basic understanding of the structure of the SSDF
    - high-level “vocabulary”
- Identify which aspects of the SSDF correspond to work products
  - interact between various groups of people
    - Managers, Administrators, Programmers

Agenda

- Introduction
- Part 2: 40,000 ft view
  - “Vocabulary” Lesson (Distilled)
    - Reading assignment (if you’re inclined)
  - Example
- Part 3: Discussion / QA

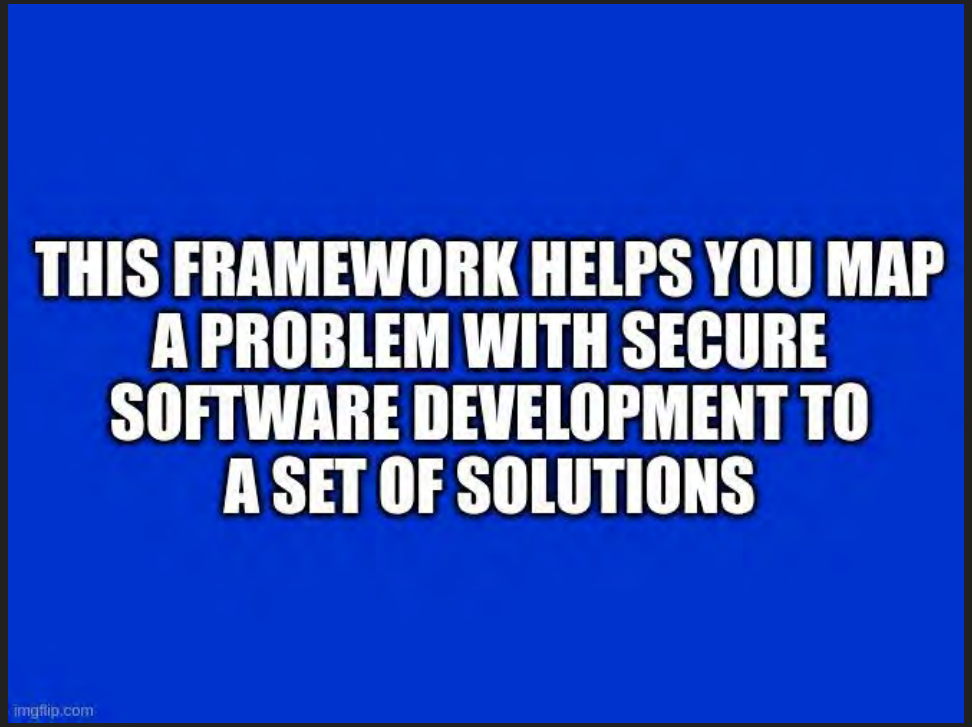
Introduction

The NIST Secure Software Development Framework

- The Secure Software Development Framework (SSDF) is a set of fundamental, sound, and secure software development practices based on established secure software development practice documents from organizations such as BSA, OWASP, and SAFECode (“global industry forum” … “effective software security programs”) - Secure Software Development Framework (SSDF)

- Abbreviations:
  - BSA - Business Software Alliance
  - OWASP - Open Web Application Security Project
  - SAFECode - “global industry forum” … “effective software security programs

In the form of a Jeopardy Clue:



About You: Familiarity with the SSDF?

A. What's that?
B. I Would like to find the time…
C. Rockin' it!

~~Implementation~~ Alignment

- Map a specific problem to a set of actionable items
  - Bug report of a security issue (Respond to Vulnerabilities)
  - Environment risk analysis, sensitive research environments (Prepare the Organization)
- Use as a guide to identify, prioritize, and fix weaknesses

Why The Secure Software Development Framework?

- The SSDF is good.
- The SSDF is extensive.
- The SSDF can serve as a guide.

But…

It’s not presented in a good way to get an overview.

Practices	Tasks	Notional 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.	**Example 1**: Define policies for securing software development infrastructures and their components, including development endpoints, throughout the SDLC and maintaining that security. **Example 2**: Define policies for securing software development processes throughout the SDLC and maintaining that security, including for open-source and other third-party software components utilized by software being developed. **Example 3**: Review and update security requirements at least annually, or sooner if there are new requirements from internal or external sources, or a major security incident targeting software development infrastructure has occurred. **Example 4**: Educate affected individuals on impending changes to requirements.	BSAFSS: SM.3, DE.1, IA.1, IA.2 BSIMM: CP1.1, CP1.3, SR1.1, SR2.2, SE1.2, SE2.6 EO14028: 4e(ix) 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-1, SA-8, SA-15, SR-3 SP800160: 3.1.2, 3.2.1, 3.2.2, 3.3.1, 3.4.2, 3.4.3 SP800161: SA-1, SA-8, SA-15, SR-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.	**Example 1**: 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 acquirers with the secure deployment, operation, and maintenance of the software. **Example 2**: 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, responses to vulnerabilities discovered in released software). **Example 3**: 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. **Example 4**: Define policies that specify what needs to be archived for each software release (e.g., code, package files, third-party libraries, documentation, data inventory) and how long it needs to be retained based on the SDLC model, software end-of-life, and other factors. **Example 5**: 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. **Example 6**: Review all security requirements at least annually, or sooner if there are new requirements from internal or external sources, a major vulnerability is discovered in released software, or a major security incident targeting organization-developed software has occurred. **Example 7**: Establish and follow processes for handling requirement exception requests, including periodic reviews of all approved exceptions.	**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**: SM1.1, SM1.4, SM2.2, CP1.1, CP1.2, CP1.3, CP2.1, CP2.3, AM1.2, SFD1.1, SFD2.1, SFD3.2, SR1.1, SR1.3, SR2.2, SR3.3, SR3.4 **EO14028**: 4e(ix) **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-8, SA-8(3), SA-15, SR-3 **SP800160**: 3.1.2, 3.2.1, 3.3.1 **SP800161**: SA-8, SA-15, SR-3 **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]	**Example 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. **Example 2**: Define security-related criteria for selecting software; the criteria can include the third party's vulnerability disclosure program and product security incident response capabilities or the third party's adherence to organization-defined practices. **Example 3**: Require third parties to attest that their software complies with the organization's security requirements.	**BSAFSS**: SM.1, SM.2, SM.2-1, SM.2-4 **BSIMM**: CP2.4, CP3.2, SR2.5, SR3.2 **EO14028**: 4e(vi), 4e(ix) **IDASOAR**: 19, 21 **IEC62443**: SM-9, SM-10 **MSSDL**: 7 **NISTCSF**: ID.SC-3 **OWASPSAMM**: SR3-A

Why This Presentation? (2)

- I can’t read the NIST SP 800-218 document
- I can read what my script generated.
  - Reasonable screen with reasonable resolution
  - Printable (yes, that)
  - Cell phone

There is a wall of text ahead.

You don’t need to read it now. You can read it whenever.

The point is that it is legible.

Prepare the Organization (PO)

PO.1: Define Security Requirements for Software Development

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.2: Implement Roles and Responsibilities

Ensure that everyone inside and outside of the organization involved in the SDLC is prepared to perform their SDLC-related roles and responsibilities throughout the SDLC.

PO.3: Implement Supporting Toolchains

Use automation to reduce human effort and improve the accuracy, reproducibility, usability, and comprehensiveness of security practices throughout the SDLC, as well as provide a way to document and demonstrate the use of these practices. Toolchains and tools may be used at 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.

40,000ft View

The Bird’s-Eye View is Generic

The high-level view of the Secure Software Development Framework isn’t really limited to Software Development.

They are good ideas in general: Guiding principles that are used for many NIST frameworks

SSDF - Definitions

- Practice Groups:
  - Prepare the Organization (PO)
  - Protect the Software (PS)
  - Produce Well-Secured Software (PW)
  - Respond to Vulnerabilities (RV)
- Practice Groups have Practices.
- Practices have Tasks.

Reading the Slides

It might help to:

- Read the slides once through
- Think about what happens when you DON’T follow the advice

The SSDF Distilled (page 1)

Prepare the Organization (PO)

- Define Requirements
- Implement Roles and Responsibilities
- Implement Toolchains (Automation)
- Define and Use Criteria for Checks
- Implement and Maintain Environments

Protect the Software (PS)

- Unauthorized Access and Tampering
- Integrity Verification
- Archive and Protect Each Software Release

The SSDF Distilled (page 2)

Produce Software (PW)*

- Design
- Review Design
- Reuse (Modularity!)
- Create
- Improve Build Process
- Review Code
- Test
- Secure by Default

* “Produce Well-Secured Software”

Respond to Vulnerabilities (RV)

- Identify and Confirm
- Assess, Prioritize, and Remediate
- Analyze to Find Root Causes

If you know those 2 slides…

If you’re looking to read something then I’d read:

- Practice Groups and Descriptions - 2 pages

Key Takeaway 1:

If you have a problem with software development that relates to security

THEN

you can use your SSDF vocabulary to classify the problem and find solutions.

Key Takeaway 2:

Don’t make doing stuff in the SSDF difficult.

Use it as a guide for what to tend toward.

Looking Back at the SSDF

- The bullet points are the left-most column (Practices)
- The reading assignment contains information from the left 2 columns (Practices and Tasks)
- There's another document (NIST.SP.800-218.SSDF-table) which has everything

Practices	Tasks	Notional 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.	**Example 1**: Define policies for securing software development infrastructures and their components, including development endpoints, throughout the SDLC and maintaining that security. **Example 2**: Define policies for securing software development processes throughout the SDLC and maintaining that security, including for open-source and other third-party software components utilized by software being developed. **Example 3**: Review and update security requirements at least annually, or sooner if there are new requirements from internal or external sources, or a major security incident targeting software development infrastructure has occurred. **Example 4**: Educate affected individuals on impending changes to requirements.	BSAFSS: SM.3, DE.1, IA.1, IA.2 BSIMM: CP1.1, CP1.3, SR1.1, SR2.2, SE1.2, SE2.6 EO14028: 4e(ix) 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-1, SA-8, SA-15, SR-3 SP800160: 3.1.2, 3.2.1, 3.2.2, 3.3.1, 3.4.2, 3.4.3 SP800161: SA-1, SA-8, SA-15, SR-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.	**Example 1**: 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 acquirers with the secure deployment, operation, and maintenance of the software. **Example 2**: 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, responses to vulnerabilities discovered in released software). **Example 3**: 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. **Example 4**: Define policies that specify what needs to be archived for each software release (e.g., code, package files, third-party libraries, documentation, data inventory) and how long it needs to be retained based on the SDLC model, software end-of-life, and other factors. **Example 5**: 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. **Example 6**: Review all security requirements at least annually, or sooner if there are new requirements from internal or external sources, a major vulnerability is discovered in released software, or a major security incident targeting organization-developed software has occurred. **Example 7**: Establish and follow processes for handling requirement exception requests, including periodic reviews of all approved exceptions.	**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**: SM1.1, SM1.4, SM2.2, CP1.1, CP1.2, CP1.3, CP2.1, CP2.3, AM1.2, SFD1.1, SFD2.1, SFD3.2, SR1.1, SR1.3, SR2.2, SR3.3, SR3.4 **EO14028**: 4e(ix) **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-8, SA-8(3), SA-15, SR-3 **SP800160**: 3.1.2, 3.2.1, 3.3.1 **SP800161**: SA-8, SA-15, SR-3 **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]	**Example 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. **Example 2**: Define security-related criteria for selecting software; the criteria can include the third party's vulnerability disclosure program and product security incident response capabilities or the third party's adherence to organization-defined practices. **Example 3**: Require third parties to attest that their software complies with the organization's security requirements.	**BSAFSS**: SM.1, SM.2, SM.2-1, SM.2-4 **BSIMM**: CP2.4, CP3.2, SR2.5, SR3.2 **EO14028**: 4e(vi), 4e(ix) **IDASOAR**: 19, 21 **IEC62443**: SM-9, SM-10 **MSSDL**: 7 **NISTCSF**: ID.SC-3 **OWASPSAMM**: SR3-A

Example(s)

Let’s Make a Generic Organization

- “Centralized” Information Technology Group
  - The organization might have more control over this group
- Research programmers
  - The Frontier / Wild Wild West
- Both groups work on similar projects but in different ways

DataSpade - Shoveling Bits Is What We Do

- Grabs a subset of data from somewhere
- Puts the data somewhere else
- Names things appropriately

BUT

- There’s a bug!
  - A specially crafted field in the data can lead to arbitrary code execution

Relevant SSDF Sections

(It’s all relevant, BUT):

- Respond to Vulnerabilities (RV) (Similar to Incident Response)
  - Identify and Confirm (RV.1)
  - Assess, Prioritize, and Remediate (RV.2)
  - Identify Root Causes (RV.3)
- Results:
  - Bug confirmed
  - Impact is limited (Hopefully…)
  - Could have been prevented by using a code analyzer
- PO.3 - Implement Supporting Toolchains
  - Code scanning could have prevented the bug
- PW.4 - Reuse Functionality
  - Apply what we learned to other projects

Set this stuff up first!

Basic Ingredients?

Generic Stuff You Need (Quick Wins!)

- Revision Control Systems (gitlab, github, self-hosted…)
- Backups
- Ticketing system (RT, Jira)
  - Provides unique identifiers to reference in commits, branches, etc (ticket/reference numbers)
  - Reasonings / journaling
- Knowledge Base, Wiki, etc (Mediawiki, Confluence)
  - Documents the environment. People really should document more.

These span multiple topics in the SSDF.

You want these to be “carrots”, not “sticks”.

Where are you with basic ingredients?

A. Just starting
B. We have the basic ingredients but not “pushed to edges”
    a. Researchers, and the departments that do things that aren’t generally known by central IT
C. Full use

Q/A , Discussion

- All materials for this talk can be found here:
  - people.ias.edu/~mvanwinkle
  - github.com/mvanwinkleias/talks
- Reading - 2024-03-NIST_SSDF_Distilled
  - NIST_SSDF_Distilled-reading_assignment_1.md - The ~2 page one
  - NIST.SP.800-218.SSDF-table.md - The BIG one

Marty VanWinkle (mvanwinkle@ias.edu)

Jay Gallman (jay.gallman@duke.edu)