Amjad Alqahtani
Assignment 3
Intro to CyberSecurity

Summary of the Document (Max 350 words, 2-Points)

The document was developed by an Executive Order numbered 14110. Basically, the EO hands the responsibility to NIST in order to create a standard document and practice for the secure AI. This is because the AI field is new and lacks standardization. So, this order/initiative addresses the urgent need for standards in the AI field, which is evolving quickly but lacks consistent guidelines.

NIST creates a community profile for the AI-specific Secure Software Development Framework (SSDF). The profile represents an extension of the general SSDF framework, but it addresses unique security requirements for AI. The SSDF community profile has few practices, tasks, and recommendations for AI model producers, system developers, and technology acquirers.

The document addresses the entire AI development software lifecycle with four primary components:

Prepare the Organization or (PO): this is an establishment of the foundational security requirements and roles specific to AI development, ensuring organizations are equipped to manage security throughout AI's lifecycle. This is important given that so many organizations are using AI.

Protect Software or (PS): protect critical AI components like model weights, training data, and configuration parameters, from unauthorized access, tampering, and potential security breaches. This is a traditional component that directly deals with AI security.

Produce Well-Secured Software or (PW): this is an implementation of AI-specific secure development practices. PW includes data integrity verification, threat modeling, and secure coding practices to mitigate vulnerabilities during development.

Respond to Vulnerabilities or (RV): this emphasizes proactive monitoring, vulnerability assessment, and incident response strategies particularly to artificial intelligence security risks, helping organizations respond effectively to emerging threats.

All of these categories are structured with the following breakdown:

Practice: explains the practice and its benefit to secure software development.

Task: Outlines specific actions required to meet each practice.

Notional Implementation Examples: provides practical examples for task execution.

Reference: Points to relevant standards and resources for additional guidance.

This community profile indeed aligns with NIST's cybersecurity and all risk management principles, providing a complete comprehensive, risk-based framework for secure AI software development. The SSDF community profile enables organizations to apply a secure-by-design approach, proactively addressing security risks in AI systems. Finally, the document focuses on AI-specific challenges that AI model producers, system developers, and technology acquirers facing such as data integrity, resistance to adversarial attacks, and secure model versioning,

Summary PS (Max 300 words, 1 point)

As the first practice of the PS, the section provides guidelines for securing AI software components (code and data) against unauthorized access and tampering. It emphasizes protecting

all elements of an AI system which includes model weights, training data, and configuration parameters. All of these will ensure confidentiality, integrity, and availability throughout the software lifecycle.

To meet above practice, the task stated that AI components like (source, executable, config-as-code) will be in a secure storage. This secure storage will be following the principle of least privilege, which means limiting access to authorized personnel, tools, and services.

The recommendation is to securely store an AI model which includes (models, weights, pipelines) to protect their confidentiality, integrity, and availability. Another recommendation is to store reward models in a different place and limit access to model weights, potentially allowing indirect access only. Another recommendation is to conduct an AI model development within approved environments and restrict human access to model weights when feasible.

Another task is about protecting training, testing, fine-tuning, and alignment data against unauthorized access and modifications. The task is to continuously monitor confidentiality (for non-public data) and integrity, and securely store data for future use.

The recommendation is to separate storage of model weights and configurations from other data, with continuous monitoring for closed models. Apply least privilege principles and enhance security with encryption, cryptographic hashing, and other risk-proportionate methods. Provide cryptographic hashes or digital signatures for AI models, artifacts, and changes to confirm software authenticity.

Another practice is to archive and protect software releases. Archive essential files and data for each software release, including versioning for infrastructure tools. Retain documentation of the AI model's selection process, training, and preprocessing steps. Track AI

model provenance, including training frameworks and sensitive data sources. If sensitive data is involved, restrict model access accordingly and consider disclosing data provenance.

Summary of PW (Max 300 words, 1-point)

It outlines secure software development practices that address specific vulnerabilities associated with AI models. This section is under the Secure Software Development Framework (SSDF), emphasizes integrating security requirements from the design phase to mitigate risks such as data poisoning, unauthorized access, and misconfigured data pipelines.

Design and development phases of the SDLC to produce secure software should address security issues during the design phase. Software should be reviewed to ensure it meets the security requirements. This should be done by the in-house security experts. You should make plans to reuse working and secure code rather than trying to write that code from scratch. Make sure to configure your compilation and build processes to ensure good executable security

Key elements include PW.1, which is high priority and focuses on incorporating risk modeling methods such as threat and attack modeling to evaluate software risks comprehensively. This task also recommends ongoing updates to risk modeling as new model versions or derivatives emerge, as well as ensuring that AI models do not operate in critical security functions without human oversight.

PW.1.2 and PW.1.3, medium-priority tasks, involve documenting and maintaining security requirements throughout the development lifecycle and leveraging standardized security tools, such as identity and access management systems, over creating custom security features. By integrating these structured tasks, PW guarantees the secure by design approach which

minimizes vulnerabilities and enhances both development efficiency and overall software resilience. Another practice is to comply with the security requirement. This is important to meet the security requirement and the satisfaction. The task will be based on a qualified person and automation process. Other practices are for confirming testing for testing and training, reuse of well secure software. Finally, creating, reviewing and configuring processes should adhere to code practice.

Summary RV (Max 300 words, 1-point)

NIST recommends that you identify and confirm vulnerabilities in your application on an ongoing basis. You need to have a means for assessing, prioritizing, and remediating the vulnerabilities in your software. If you have several vulnerabilities in your software you need to have a means for prioritizing the most server vulnerabilities so that they can be addressed first. Fixing a vulnerability may not necessarily address the root cause of the issue. It is important to identify the root cause for each vulnerability.

For this practice, RV needs a task that continually gathers the information from a variety of sources on a potential vulnerability. The best recommended ways are using logs and monitoring tools, separate users of the model from reporting potential security issues, and monitor vulnerability and incidents. Another task is to make sure that previous vulnerability is not existing in the new version. This can be done by testing, reviewing, and analyzing. This task recommends scanning the code, automated the process, and conducts the auditing.

Another practice is to remediate the vulnerability based on the priority. Gather sufficient information about the vulnerability which can be recommended by deep analysis and input and

output to notice the deviation from normal case. Another task is to have risk responses for vulnerabilities. The recommendation is to risk response with time and expenses consideration.

A useful approach is to identify the root causes of any issues. This starts by analyzing identified risks to uncover their underlying causes. Root causes can be determined by reviewing processes such as training, testing, and fine-tuning. Over time, analyzing these causes can reveal patterns, allowing us to check the software for similar vulnerabilities, review the SDLC process, and make updates as needed. This helps to prevent or reduce the chances of the root causes recurring.