Student Name: Leeya Thercy

Point 1: Software Bill of Materials

Observation: The absence of a comprehensive Software Bill of Materials (SBOM) in software projects can lead to a lack of transparency regarding the components and dependencies within the codebase.

Importance: A Software Bill of Materials is crucial for understanding the composition of software, identifying vulnerabilities, and ensuring compliance. It provides a detailed inventory of all components used in a project, including third-party libraries and their versions.

Impact: Without a clear SBOM, developers may struggle to track and manage software components effectively. This can result in security risks, as vulnerabilities in third-party libraries may go unnoticed. Additionally, it hinders efficient collaboration and makes it challenging to address issues promptly, impacting the overall security and reliability of the software.

Point 2: Repository Trust

Observation: The trustworthiness of software repositories is paramount, as relying on insecure or compromised repositories can compromise the integrity and security of the software being developed.

Importance: The trustworthiness of a repository directly influences the reliability of the code it contains. Secure and reputable repositories help ensure that the software components and dependencies are authentic, unaltered, and free from malicious code.

Impact: Using untrustworthy repositories exposes software projects to the risk of incorporating malicious or compromised code, leading to potential security breaches and compromised system integrity. It is essential to establish and maintain a robust trust model for repositories to mitigate these risks and uphold the overall security of the software development process.

Point 3: Inherited Risks

Observation: Inheriting risks from third-party components, libraries, or frameworks is a common challenge in software development. Dependencies may introduce vulnerabilities or issues that developers inherit unknowingly.

Importance: Recognizing and addressing inherited risks is crucial for ensuring the overall security and stability of the software. Failing to assess and manage these risks can lead to unforeseen vulnerabilities and compromise the reliability of the developed software.

Impact: Neglecting to address inherited risks can result in security breaches, system failures, and increased maintenance efforts. By proactively identifying and mitigating these risks, developers can enhance the robustness of their software, reduce potential vulnerabilities, and create a more secure foundation for the development and deployment of applications.