Benjamin Sturgeon

CS-405

April 26, 2025

Module 8 Journal

The readings and assignments in this course provided an excellent to basic concepts of secure coding and standards. As I developed coding standards based on our coding standards resources, I was curious how these standards could be met in the SDLC. With the completion of documentation and examples, it is now clear that taking a thorough and careful approach so that each concept is implemented in a safe and appropriate manner. An example of one of these standards is data sanitization. This dictates that data be verified when processing into sub-systems so that a vulnerability is not created. This approach does not prescribe a single correct method but instead provides guidelines on improper implementations and associated risks. Understanding these concepts enables individuals to apply the standards effectively to improve their own code.

Security should never be an afterthought. In a project’s development cycle, complexity increases as progress is made. Delaying security implementation until the final stages makes integration significantly more challenging, requiring a thorough reevaluation of various components. If security features are incompatible with the existing architecture, backtracking becomes necessary. Conversely, incorporating security early allows for synchronized testing alongside system code, ensuring security measures align with the program’s intended functionality.

The balance between risk mitigation and implementation costs is a key factor in secure coding. While integrating security early demands additional time and expertise, these costs are outweighed by the benefits of risk mitigation. Depending on project size and the sensitivity of stored data, effective security measures can prevent costly attacks and safeguard assets such as customer trust. However, real-world projects often face time constraints and resource limitations, requiring careful risk assessment to prioritize security while meeting deadlines.

Zero-trust is increasingly vital as cloud storage and computing become standard. Implementing security measures such as reverse proxies and verifying device permissions strengthens this approach. Security policies should be structured in an organized, clear, and simple manner. The principle of 'Keep It Simple' applies broadly—streamlined policies enhance adoption, enabling stakeholders to implement security practices more effectively.

References:

*Sei cert C++ coding standard*. SEI CERT C++ Coding Standard - SEI CERT C++ Coding Standard - Confluence. (2024, February 14). https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88046682

Mikejo5000. (n.d.). *Use C++ assertion statements for debugging - visual studio (windows)*. Use C++ assertion statements for debugging - Visual Studio (Windows) | Microsoft Learn. https://learn.microsoft.com/en-us/visualstudio/debugger/c-cpp-assertions?view=vs-2019