Alexander DeMarco

SNHU CS-405 Module 8 Journal

Professor Ramsey Kraya

04/27/2025

**Adoption of a Secure Coding Standard**

Throughout the course, I gained a deeper understanding of secure coding standards and how to apply them throughout the development life cycle. Leaving security to the end only creates vulnerabilities that are more difficult and expensive to track down and fix. It can also result in a loss of credibility for a company if a breach occurs. Standards such as the SEI CERT C++ guidelines provide a framework to follow while designing a system, helping eliminate known vulnerabilities and ensuring that best practices are followed from the ground up. Applying secure coding early and proactively reduces the number of attacks surfaces and prevents many common mistakes before they become serious issues.

**Evaluation and Assessment of Risk and Cost Benefit of Mitigation**

One key takeaway from the course is understanding how to evaluate the risk of vulnerabilities and assess the cost-benefit of mitigation strategies. Not all vulnerabilities carry the same weight, and not every fix is worth the potential cost if it introduces new risks or complexity. Using threat matrices to assign risk values allowed me to prioritize vulnerabilities and focus efforts on mitigating the most critical risks before deployment. Addressing high-risk threats, such as buffer overflows or authentication failures, first ensures that system stability and security are preserved without wasting resources on low-impact issues.

**Zero Trust**

The concept of Zero Trust shifts priority toward securing the end user, especially due to the widespread use of mobile devices. It is much more difficult to trust devices when they are scattered across the world and accessing internal systems remotely. Implementing Zero Trust strategies involves strong authentication mechanisms, continuous monitoring, and enforcing the principles of least privilege across all systems. The Zero Trust model creates a decentralized form of defense where identity verification becomes the foundation of system access, significantly reducing the risk of insider threats or compromised devices. This shift from implicit trust to continuous verification represents a major evolution in security architecture, addressing the realities of modern distributed environments.

**Implementation and Recommendations of Security Policies**

The implementation and enforcement of security policies are critical for all systems. Policies must be actionable and enforceable across all stages of development, including coding practices, unit testing, static code analysis, and DevSecOps integrations. Tools like SonarQube and cpluspluscheck improve security by automatically identifying vulnerabilities and helping expand validation coverage early in the development process. Security should not be treated as a final checklist item, as it often has been in the past. Instead, it must be front and center in system design, continuously monitored, tested, and maintained throughout the entire development life cycle.