**Module Six Journal**

Joseph Veneski

Department of Computer Science, Southern New Hampshire University

CS-405 Secure Coding

Professor Kaan Esendemir

April 14, 2024

**Module Six Journal**

**Security Up Front:**

Security is a critical part of the Software Development Life Cycle (SDLC) but often it has been left to the testing phase with minor superficial security testing if at all (Red Hat, 2022). The phrase “Don’t leave security to the end” emphasizes a key best practice in incorporating security throughout the SDLC, not just at the end during testing and maintenance. Early adoption of security can mitigate many costly redesigns later in a project, as well as lead to more secure overall applications with less likelihood of bad actors compromising and incurring reputation and financial damages. There is no way to prevent every type of bug or vulnerability but a security mindset and utilizing a comprehensive security policy can help create much more secure applications and align with any security regulations that may apply. In a study by IBM’s System Sciences Institute, it was found that the cost to remediate bugs identified in testing rather than the design phase cost up to fifteen times as much (Dawson et al., 2010).

**Preventing Threats:**

There are several steps that can be taken to aid in preventing potential threats. As stated previously, incorporating security early on into development is beneficial. Security requirements should be identified right along with functional requirements. What is the potential threat level for the system being designed, common vulnerabilities or bad actors to be aware of? Establishing a comprehensive security policy for teams to follow can help serve as a guide as well as helping ensure alignment on standards ensuring security is everyone’s concern. A common pattern in all software development is the need to continuously learn and adapt to new technologies and such is the same with security. It is greatly beneficial to stay abreast of the latest in security threats and best practices as they evolve over time.

**Example of Security:**

To ensure that security is a high priority in the project security requirements will be established early along with functional requirements including key topics such as authentication/authorization and what sort of encryption to use. Including these concerns early in planning ensures they are not left behind and are integrated as development unfolds. In addition to establishing security requirements from project inception, use of dependency management and static testing tools will be utilized to further scan for potential vulnerabilities. Tools utilized could include but not be limited to CPPCheck, NPM Audit, and OWASP Dependency Check (OWASP, 2024).

**References**

Dawson, M., Burrell, D. N., Rahim, E., & Brewster, S. (2010). Integrating Software Assurance Into The Software Development Life Cycle (SDLC). *Journal of Information Systems Technology & Planning 3*(6), 49-53. <https://www.researchgate.net/publication/255965523_Integrating_Software_Assurance_into_the_Software_Development_Life_Cycle_SDLC>

OWASP. (2024). *Vulnerable Dependency Management Cheat Sheet.* <https://cheatsheetseries.owasp.org/cheatsheets/Vulnerable_Dependency_Management_Cheat_Sheet.html>

Red Hat. (2022). *Security in the Software Development Lifecycle.* <https://www.redhat.com/en/topics/security/software-development-lifecycle-security>