Module 8: Journal – Portfolio Reflection

CS-405 Secure Coding

Zachariah Garrison

Prof. Tam

08/24/2025

As I have discussed from the start of this course, security is a priority and should always be treated as such. Some development practices rely on an applying security in the end or as a patch in later, though security should be involved from the beginning stages in planning and be implemented through every stage and continue to be a focus even after the application is rolled out.

**Adoption of a secure coding standard, and not leaving security to the end**

As stated before adopting a secure coding standard is essential to prevent vulnerabilities early in the software development lifecycle. There exists standards that could be followed such as OWASP and CERT Secure Coding Standards that place emphasis on writing code that mitigates common vulnerabilities such as SQL injections and buffer overflows. OWASP recommends input validation and sanitization to ensure data integrity, while CERT advocates for proper error handling to avoid information leakage. By placing emphasis on secure coding practices from the start, rather than treating security as an afterthought, reduces the chance of vulnerabilities making it through, possibly costing expensive post development fixes. An article on HackerOne discusses the cost difference between finding and fixing vulnerabilities in development vs production, “Even when a patch involves an update to just a few lines of code, determining what lines of code can be like looking for a needle in a haystack. Fixing vulnerabilities discovered in production is roughly 30 times more expensive than finding and fixing them during development” (Mateer, 2025). Mateer also goes on to discuss that at a labor rate of $100 per hour that post-production fixes will be roughly $1500, while found in pre-production and development phases would be closer to $50. Clearly showing that by having a security focused plan of development not only creates better more secure code, but can be potentially financially beneficial.

**Evaluation and assessment of risk and cost benefit of mitigation**

Risk assessment is paramount in having an effective security strategy, taking the time to identify, prioritize, and mitigation of potential threats. By focusing on An example being a high likelihood, high impact vulnerability, such as an unpatched remote code execution flaw, demands immediate attention, while lower risk issues may be deferred based on cost benefit analysis. The cost of mitigation includes resource for implementing controls, such as encryption or access controls, versus the potential financial (but also reputational) damage of a breach. The Equifax data breach exposed sensitive data of millions of Americans and can show the cost of not properly weighting the cost of risk management. A cost benefit analysis might justify investing in automated tools that would scan for vulnerabilities or to further test applications for any gaps or holes that could be accessed, as these up-front costs would be minor compared to full scale breach expenses. By taking time to prioritize risks based on impact and likelihood, you can determine how to allocate resources to create a secure application with the funds attached to the project.

**Zero trust**

The zero trust model operates on the policy of “never trust, never verify” which objectively rejects that any device, user, and network are secure. Zero trust focuses on requiring continuous authentication and least privilege access. This approach is becoming more common practice as many companies continue to adopt this theory (even SNHU has begun requiring two factor authentication every ten days). Implementing zero trust polices can prevent accidents, a report I did for a former class focused on the impact of the Colonial Pipeline ransomware attack and how compromised credentials were the cause of that incident. “In May of 2021, a hacker group known as DarkSide gained access to Colonial Pipeline’s network through a compromised VPN password. This was possible, in part, because the system did not have multifactor authentication protocols in place” (INSURICA, 2024). I apologize as I cannot seem to find the source but want to highlight that the incident occurred from an employee having their work information open at a coffee shop, while I hate not verifying this I want to emphasize that through proper two set authentication could have prevented this information from being accessed and used in the ransomware attack and as a note that no location is to be deemed safe. By adopting a zero trust policy, along with secure coding practices, would limit exposure to sensitive information.

**Implementation and recommendations of security policies**

Having effective security guidelines and polices in place help create consistent, secure, and proper code. These policies should focus on secure coding standards, regular risk assessment, and adhere to good Defense in Depth and Zero Trust strategies. By combining the two policies, you can create a robust foundation in which all training could be based, and all development work is built on. I would recommend that all team members be trained on a set policy designed between the two policies and complete training exercises. All team members should be familiar with OWASP and CERT as these should be integral parts of the security policy and provide good reference to refer to when needed. These policies should also enforce regular code reviews, include automated testing, and comply with all standards and regulations. By following the outlined policies in Defense in Depth and Zero Trust, organizations could adopt DevSecOps practices, and have security in CI/CD pipelines to automate vulnerability scanning and ensure compliance is met. It should also be noted that even though well-defined security policies be designed and implemented, a well-rehearsed and planned incident response plan should be created to address any breach quickly. By having this incident response plan, organizations should be able to reduce the amount of damage cause if a breach was to occur, and should be able to eliminate said threat swiftly.

**References**

**INSURICA. (2024). *Cyber case study: Colonial pipeline ransomware attack*. INSURICA. https://insurica.com/blog/colonial-pipeline-ransomware-attack/**

**Mateer, D. (2025, February 25). The Cost Savings of Fixing Security Flaws in Development [Review of The Cost Savings of Fixing Security Flaws in Development]. HackerOne.** [**https://www.hackerone.com/blog/cost-savings-fixing-security-flaws**](https://www.hackerone.com/blog/cost-savings-fixing-security-flaws)

**‌**