**8-2 Journal: Portfolio Reflection**

Winnie Kwong

Southern New Hampshire University

CS-405: Secure Coding

Professor Tam

August 25th, 2024

**Secure Coding Standards, Not Left at the End**

Adopting the secure coding standard and not leaving security at the end has increased my awareness and expanded my mind to prioritize security as an integral part of development. I learned that establishing a robust coding standard early on can reduce the potential risks and costs of neglecting security. According to Murray, data breaches are reoccurring because of poor coding. (Murray, 2020). DevSecOps can be used to enforce and comply with coding standards to create a more robust and secure software development lifecycle. DevSecOps can be implemented by using automation throughout the pre-production and production phases to ensure that security, efficiency, and reliability can be enhanced in each stage. The only unresolved issue is understanding how to balance security and development speed. Although finding the balance can be used with automation and continuous monitoring, other specific challenges, such as infrastructure differences and overly complex standards, can interfere.

**Evaluating and Assessing Risks and Cost**

Finding a balance between security and functionality relies on evaluating and assessing risk and cost benefit to mitigate vulnerabilities that can lead to security breaches, such as SQL injections. I learned it requires understanding potential threats, impacts, and resources to implement technical and strategic coding. According to Vijayan, Static Application Security Testing (SAST) tools enable developers to identify and address potential weaknesses. (Vijayan, 2022). Tools like CPPcheck and SonarQube can analyze and detect buffer overflows and issues handling insecure data. They are valuable tools for identifying undefined behaviors in C and C++ code. Although the course has provided a comprehensive overview, further exploration is needed to underscore the importance of proactive risk management.

**Zero Trust**

Zero trust has significantly shifted my cybersecurity perspective to enhance my knowledge and understanding of vulnerabilities. The additional reading in the modules made me want to learn more about core principles, their purposes in modern security architecture, and the zero trust model because it highlights the complexity of cyber threats and provides an alternative to traditional network security models. Traditional models meant all systems and users were built on implicit trust but have proven vulnerable. In contrast, with zero trust, it mandates continuous verification of every user and device to mitigate breaches and broaden past securing only networks. Adopting the mindset of “trust but verify” to “never trust, always verify” helps transition from traditional to modernized security methods. (Kueh, 2020). Embracing the zero trust model makes it possible to address vulnerabilities through explicit verification, least privilege access, micro-segmentation, and continuous monitoring to reinforce the cybersecurity framework further. Acknowledging the reality of network threats has influenced me to measure the protection of sensitive data and systems proactively.

**Security Policies**

Security policies impacted my understanding of their overall security posture. Security policies are a living framework that guides behavior and safeguards sensitive information. I have learned that developing comprehensive security is just the start and needs constant reviews and updates to combat emerging threats. My experience with the unit testing assignment helped me understand the challenges of developing a secure code while maintaining efficiency, simplicity, and readability to establish a more secure software system. According to Dunham, developers should research security at least yearly and update sooner as needed. (Dunham, 2020). Emphasizing the importance of keeping policies current can enable organizations to handle new security challenges and protect sensitive information.

**Conclusion**

As technology rapidly evolves, the threat landscape demands innovative approaches to cybersecurity. When the next generation of cybersecurity emerges, we are responsible for proactively identifying new strategies to stay one step ahead of hackers. Throughout the course, we explored several crucial concepts that form the foundation of secure coding. These include adopting a secure coding standard, integrating security from the beginning rather than as an afterthought, evaluating and assessing risk and the cost-benefit of mitigation strategies, implementing a zero trust model, and developing and recommending effective security policies. As the future of cybersecurity unfolds, it is our duty as developers to stay vigilant, adaptable, and committed to continuous learning.

**References**

Dunham, R. (2020, May 5). The importance of a company information security policy. *Linford & Company LLP*. <https://linfordco.com/blog/information-security-policies/>

Kueh, T. (2020, January 15). *A practical guide to Zero-Trust security*. Threatpost; Tony Kueh. Retrieved August 19, 2024, from <https://threatpost.com/practical-guide-zero-trust-security/151912/>

Murray, A. (2024, July 26). Secure Coding: A Practical Guide. *Mend*. <https://www.mend.io/blog/secure-coding/>

Vijayan, J. (2022, November 16). 6 DevSecOps best practices: *Automate early and often*. TechBeacon. <https://web.archive.org/web/20240117143629/https://techbeacon.com/security/6-devsecops-best-practices-automate-early-often>a