Gracie Ream

CS 405

8/24/2025

8-2 Journal: Portfolio Reflection

Throughout this course I have gained knowledge pertaining to the importance of practicing secure coding and security principles. Security should not be left as an afterthought but rather be implemented in various stages and phases of the development life cycle.

Following the secure coding standards, given by OWASP, we can mitigate regular occurring security vulnerabilities. With the SEI CERT secure coding standards, we can follow and implement these secure coding standards to grant even more security within an application. By writing secure code, the risk of SQL injection, cross-site scripting and unsecure authentication can be eliminated or decreased. By integrating these guidelines and principles early on and all throughout the development life cycle, we can ensure our application is secure and safe. This will reduce the risk of multiple vulnerabilities showing up later on in the development process which results in extra cost and time to fix them later. “Security starts with your code, and creating secure code is a vital part of creating a great software product” (Morrow, 2023).

This course not only evaluated the importance of secure coding standards but emphasized the process of evaluating risks and assessing them. It is best to prioritize based on cost, likelihood and impact for vulnerabilities. Lower risk vulnerabilities can be managed with monitoring and logging rather than full mitigation. This reduces costs but still keeps the application secure. Adding encryption for every data transfer can increase overhead but is necessary for sensitive information. By weighing risks and costs together, we can recommend solutions that are balanced and assess the risks effectively.

Moving onto the zero-trust principle, this implies that no device, user, data, network or system is automatically trustworthy. By shifting the security from a perimeter-based model to one that favors least privilege and enhances authentication and verification, we can implement a more secure environment. This means making applications that incorporate multi-factor authentication, role-based access control and monitoring/logging methods. Once trust is established across each one, access can be granted or denied and once the access is granted, it is best to constantly reverify who or what has access. (Kueh, 2020). Overall, by implementing these security measures and automatically assuming nothing can be trusted, we create a better and more secure place.

This course greatly deepened my understanding of how to guide developers and organizations on what they should be incorporating into their frameworks. A framework that follows security best practices like least privilege, defense in depth and the AAA policy as well as code reviews is one that I would recommend. I would recommend policies that enforce secure coding standards and reviews before deployment and require static analysis tools to be used to catch vulnerabilities. Embedding tests in phases of the DevSecOps Pipeline is also important because code errors will be caught before deployment. Multiple layers of security in each development phase ensures a more secure system, and using authentication, authorization and accounting techniques, organizations can mitigate risk more effectively. Overall, by using these practices, the organization can develop a standard on how they can create a better, maintainable and a maximized secure environment.

**References**

Kueh, T. (2020, January 15). *A Practical Guide to Zero-Trust Security*. Threatpost.com. <https://threatpost.com/practical-guide-zero-trust-security/151912/>

Morrow, S. (2019, October 18). *What is Secure Coding and Why is It Important?* VPNOverview.com. <https://web.archive.org/web/20240414021338/https://vpnoverview.com/internet-safety/business/what-is-secure-coding/>