8-2 Journal: Portfolio Reflection

Gavin M. Bish

Southern New Hampshire University

CS405: Secure Coding

Professor Philomena Ogoh

February 24th, 2025

8-2 Journal

Throughout this course, one of the dominant themes has been the understanding that security cannot be treated as an afterthought. The adoption of a secure coding standard underscores this priority by positioning security principles at the forefront of the software development lifecycle. As illustrated by the OWASP Top Ten (OWASP, 2021), a “shift-left” approach helps developers address potential vulnerabilities early in the design and coding phases rather than scrambling to fix them once an application is nearly complete. By prioritizing secure coding from the start, organizations reduce the risk of costly rework, create more stable products, and cultivate a broader culture of security awareness among development teams.

Evaluation and assessment of risk have also played a critical role in our discussions, especially in balancing cost against the benefits of implementing robust security measures. In practice, every organization must weigh the likelihood and potential impact of a breach against the expenses incurred by proactive defensive strategies. As various case studies throughout the course demonstrated, risk management is an iterative process that demands ongoing assessments at each phase of software development. By assigning quantifiable metrics to the severity of vulnerabilities, security teams can efficiently allocate resources and address the most pressing risks without overburdening developers or undermining project timelines.

Zero trust, introduced as the principle that “no one is safe,” challenges traditional notions of perimeter security. Instead of granting blanket trust once a user or system has passed an initial check, zero trust enforces continuous verification of identity, context, and device posture (National Institute of Standards and Technology [NIST], 2020). This approach—often implemented through micro-segmentation—limits lateral movement within a compromised network and ensures that even internal services require authentication and authorization for each request. From a developer’s perspective, building applications that support a zero trust model can reinforce the principle of least privilege, effectively reducing the attack surface by ensuring that users and services have only the minimal access needed to perform their tasks.

Finally, the implementation and recommendation of security policies tie all these elements together. Well-crafted policies define how a secure coding standard should be adopted, outline parameters for continuous risk assessment, and specify how zero trust principles should be enforced across an organization’s infrastructure. From the readings on governance and compliance, it is clear that policies must remain living documents that adapt to evolving threats. They should be supported by thorough training and leadership endorsement, reinforced by automated tools integrated into continuous integration and delivery pipelines. By blending these principles into a comprehensive strategy—from secure coding standards to rigorous risk evaluation, zero trust architectures, and dynamic security policies—organizations can significantly improve their ability to safeguard systems and data in an ever-changing digital environment.

**References**

National Institute of Standards and Technology. (2020). *Zero Trust Architecture (NIST Special Publication 800-207)*. <https://doi.org/10.6028/NIST.SP.800-207>

OWASP Foundation. (2021). *OWASP Top Ten 2021*. <https://owasp.org/www-project-top-ten/>