**8-2 Journal: Portfolio Reflection**

The following topics are discussed as a synthesis of course concepts.

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

Adopting a secure coding standard means implementing security *throughout* the software development lifecycle (SDLC) rather than toward the end of it. One example of this is thorough unit testing for source code, which can reveal vulnerabilities or errors before they’re exploited at launch (Morrow, 2023). The sooner risks are weeded out and resolved, the less expensive they are in terms of effort, reputation, and budget (Morrow, 2023). Also, security doesn’t just refer to malicious activity. Human error can damage a project or system, and adopting a secure coding mindset throughout development also works to mitigate this. It’s about being conscientious of anything that could go wrong and how to mitigate or eliminate that risk.

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

Risks are evaluated for their likelihood to occur and assessed for their impact if exploited (Nagarajan, 2024). Identifying weaknesses early reduces costs by mitigating damage done in an attack (Nagarajan, 2024). This goes along with zero trust (discussed below) in that assuming attacks *will* happen and planning accordingly makes software more secure overall. The benefit of assessing risks and implementing prevention tactics early and often is that it becomes a natural part of development rather than a scramble to figure out after an attack occurs.

**Zero trust**

Zero trust is the mindset that external defenses, like a firewall, *will* fail and that the base layers of a system must be made to withstand threats on their own (Kerman, 2020). An outer layer, like a firewall, does nothing if an attack comes from within, such as through a compromised account (Kerman, 2020). To put zero trust in practice, developers must verify rather than assume that a network or subsystem is safe from threat (Kerman, 2020).

**Implementation and recommendations of security policies**

Security policies exist to offer a guiding hand for development teams. The one written for this course covered well-regarded principles for secure coding and created coding standards from them. Coding standards in a security policy show how implementation works for different areas and different principles via compliant code examples. They also provide recommendations on where to prioritize efforts and which automation tools are suited for catching different vulnerabilities. Security policies also act as a reference for the shift from DevOps to DevSecOps, or more generally a shift into a culture of secure coding (Casey, 2021). It helps to have a guide to look back to as teams make these shifts and go through training, just like it’s helpful to have documentation on hand when learning a new programming language.

References

Casey, K. (2021, October 13). *How to create an effective security policy: 6 tips.* The Enterprisers Project. <https://enterprisersproject.com/article/2021/10/security-policy-how-to-create>

Kerman, A. (2020, October 28). *Zero trust cybersecurity: ‘Never trust, always verify’.* National Institute of Standards and Technology. <https://www.nist.gov/blogs/taking-measure/zero-trust-cybersecurity-never-trust-always-verify>

Morrow, S. (2023, August 22). *What is secure coding and why is it important?* VPN Overview. <https://web.archive.org/web/20240414021338/https://vpnoverview.com/internet-safety/business/what-is-secure-coding/>

Nagarajan, J. (2024, March 27). *How to perform a cybersecurity risk assessment.* CrowdStrike. <https://www.crowdstrike.com/en-us/cybersecurity-101/advisory-services/cybersecurity-risk-assessment/>