8-2 Journal: Portfolio Reflection

This course has provided an invaluable perspective on the role security plays throughout the software development lifecycle and the best practices a developer can follow to write secure code. While the probability of a given data breach might be low, it does not mean that malicious actors are not actively trying to find a weak spot to infiltrate through. Security in software includes understanding the various options and best practices available, practical implementations and frameworks, and proactive mentality required to remain vigilant.

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

A secure coding standard goes beyond just aligning everyone to best practices to follow and be aware of. It allows the team to identify and address vulnerabilities earlier into development and alleviates the need for significant refactoring as more edge cases are covered and tested. Whether it’s the SEI CERT as we used here, OWASP, or CWE, teams should try to consider a coding standard that can help address common security issues early on. This notion of not leaving security to the end enforces the idea of security as a shared responsibility from everyone across the DevSecOps pipeline. Otherwise, teams are likely to defer the validation of their work to testing and QA teams that could result in significant, preventable refactoring late into development. (Foster, 2020)

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

Risk is a constant for any application and organization. Risk assessment takes a considerable amount of time and resources that directly impact the associated cost to ensure mitigation. More complex systems may require more complex analysis to evaluate potential vulnerabilities and the associated priorities and costs to correct them, but even simpler projects must strike a balance between budgetary constraints and security measures. Overall, the cost of mitigation often outweighs the cost of implementation as companies might have to bear more than just loss of data but also loss of reputation and clients.

**Zero trust**

The prominence of cloud architecture and work from home policies have extended the potential entry points and surface area for a given threat. Since the threat model has changed, the security model has also had to adapt. Zero trust is an absolute framework for enforcing security at all external and internal levels within a system or network. Unlike traditional approaches to security that forgo authenticating a user beyond a certain barrier, zero trust policies assume that nothing and no one can be trusted and require repeated verification. (Kueh, 2020) This might appear inconvenient with the need for key cards, multi-factor authentication, or inactivity timeouts but the policy ensures that risk is proactively minimized and mitigated where possible.

**Implementation and recommendations of security policies**

The implementation of a security policy depends on the quality of the policy itself. It is easier to follow a well-written policy that does not leave room for ambiguity or interpretation as all individuals will have clear guidelines and expectations to reference. Consistency can be attained as teams find clarity in the recommendations provided by the security policy. Effective security, however, relies on the proactive evolution of the policy to ensure that it meets the changing needs of an organization and threats as they change and arise.

**References**

Foster, S. (2020, October 22). *Security standards: What are secure coding standards?*. Perforce Software. https://www.perforce.com/blog/qac/secure-coding-standards

Kueh, T. (2020, January 17). *A practical guide to zero-trust security*. Threatpost English Global threatpostcom. https://threatpost.com/practical-guide-zero-trust-security/151912/