**8-2 Journal: Portfolio Reflection**

Walter H. Briones

Department of Computer Science, Southern New Hampshire University

CS-405: Secure Coding

Professor Alan Spencer (M.S)

August 18th, 2025

**Portfolio Reflection**

**Adopting a Secure Coding Standard / Leaving Security to the End**

The act of adopting secure coding standards as part of the software development life cycle allows programmers to effectively deploy code that mitigates security threats and removes commonly exploited vulnerabilities (Morrow, S. 2023). A secure coding standard defines secure coding principles, coding standards (including common vulnerabilities found in all manner of coding languages), how to test for and mitigate exploitable code, security concepts, and risk assessments, among others. Such documentation is paramount to the development of secure software as it encourages teams to adhere to a defined standard based on industry best practices and serves as a reference for useful security-based activities. Furthermore, a secure coding standard establishes that an organization takes the security of their systems and digital assets seriously, further helping to entrench a culture of approaching development with a security-first mindset.

The antithesis of a security-first approach would be a “leaving security to the end” purview. Leaving security to the end means not taking the security of a system into consideration until the final phases of development. Such an approach may lead to consequences such as developing systems with deep vulnerabilities that leave it susceptible to attack. Fixing such issues generally requires excessive resources in the form of manpower, money, and time. Much more is also at stake when leaving security as a second thought. Damage to an organization’s reputation, possible legal ramifications, and loss of public trust could result from an inadequate approach to proper system security. Therefore, it is always prudent to approach the software development life cycle with a DevSecOps concept – that is to say, that security should be incorporated into the development process from the “ground-up.”

**Evaluation and Assessment of Risk / Cost Benefit of Mitigation**

Individual coding standards come with an inherent risk associated with the vulnerability being addressed. Some risks are more costly than others in terms of the likelihood of occurrence, and severity. Because of these risk dynamics associated with exploitable vulnerabilities, it is important that the security policy involves clear evaluations of associated risks, as well as a cost/benefit analysis of mitigation. Conducting such analysis helps teams prioritize which issues should be addressed first, and as efficiently as possible. As an example, an unlikely scenario involving a low severity security vulnerability may give way to a highly likely and severe security flaw that would need to be immediately remedied. This is also where cost benefits also come into consideration. It is in a team’s best interest to integrate security into their development cycles to reduce time and resources spent on security remediation (Dora, n.d.).

**Zero Trust**

Zero trust is a security policy that dictates that an organization deploys a system that treats all users and connections to a system – both internal and external – with a policy that eliminates trust (Brook, C. 2024). This means that the security policy is more stringent in its requirements for proper authentication and identity verification. Zero Trust is a model of security that assumes all attempts at access is a potential threat, thus limiting such accesses to system components down to a bare minimum. This is an example of utilizing the principle of least privilege, which allows access to only the components needed to complete a task. As a policy, Zero Trust’s advent is in response to an ever-increasing threat landscape where de-centralization of data due to cloud infrastructure integrations, societal shifts to hybrid work models, social engineering, and other potential threat vectors are becoming increasingly difficult for legacy systems and policies to keep up with.

**Implementation and Recommendations of Security Policies**

Not all systems and their components are built the same. With this being the case, implementation and recommendations of security policies heavily rely on what the organization and team are trying to accomplish with the project they are developing. However, a secure coding policy based on overarching principles, as well as a set of secure coding standards is a good first step toward implementing security. Such a policy can help an organization shift the culture from leaving security to the end, to developing with security at every juncture of the project’s progress begins with well-defined policies and priorities for development teams to follow. Assessing and re-evaluating the policy at regular intervals also ensures that the policy is just as dynamic as the vulnerabilities that pose a real threat to the system-in-progress.

**References**

Dora: Devops capabilities: Shifting left on security. DORA | DevOps Capabilities: Shifting Left on Security. (n.d.). <https://web.archive.org/web/20240524180526/https://dora.dev/devops-capabilities/process/shifting-left-on-security/>

Morrow, S. (2023, August 22). 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/>