**Journal: Portfolio Reflection**

Justin Starr

Department of STEM

CS 405 – Secure Coding

Professor Mike Alesso

August 25, 2024

**Journal: Portfolio Reflection**

The adoption of a secure coding standard is essential and should be implemented as part of a security policy to help mitigate and reduce the likelihood of vulnerabilities being introduced in code. Mitigating these potential vulnerabilities also helps to prevent exposing a companies infrastructure where an attacker might be able to gain access to various systems and conduct malicious activities, such as the theft of data, or even cause systems to stop functioning correctly. Additionally, adopting a secure coding standard helps increase the quality of products as well as resilience against cyber attacks by implementing security throughout the Software Development Lifecycle (SDLC). Oftentimes, companies following a straightforward DevOps methodology leave security till the end instead of incorporating security throughout the SDLC, which can lead to time consuming and costly side effects. Leaving security until the end increases the likelihood that mistakes will be made, which can introduce vulnerabilities in software that can go unnoticed. If vulnerabilities are discovered in software during the testing phase or even worse, during deployment, it can cost an organization not only a great amount of time, but a large amount of financial resources to remediate the issue. In some cases, a project may even have to be scrapped altogether because the organization does not have the financial resources to remediate the issue at such a late stage in the project.

It is essential that organizations evaluate and assess risks and the cost benefit of mitigation. Potential vulnerabilities should be identified as soon as possible to assess their impact and associated risks. Risks should then be prioritized based on their associated severity and how likely they are to occur. Once these risks have been prioritized, the cost of mitigating these risks should be evaluated to determine the required cost of implementing needed security measures alongside the potential losses that could arise from not taking action, such as reputational or financial damage. Benefits of acting now include the ability to prevent future attacks from occurring, cost savings associated with dealing when data breaches occur, improved product quality (production of more secure and reliable software), as well as the ability to maintain the organization’s reputation and trust with its customers.

The zero-trust framework is a security framework where trust across every user, device, and application are continuously verified and should be implemented into an organization’s security policy. This ensures that trust is continuously verified before granting only the required access by applying the principle of least privilege (Kueh, 2020). In doing so, access to an organization’s resources is limited to only what is required to accomplish a particular task and reduces the likelihood of an attacker from gaining access to these resources or executing commands with elevated permissions.

By adopting a secure coding standard, such as incorporating standards like the SEI CERT C++ Coding Standard will reduce the likelihood of introducing vulnerabilities from the start of project development. All identified vulnerabilities should be evaluated to determine their potential impact and associated risks, as well as the time and cost associated with mitigating these vulnerabilities. Prioritizing and implementing security throughout the Software Development Lifecycle helps reduce the likelihood of introducing security vulnerabilities that can lead to financial and reputational harm to an organization if security is left until the end. Security policies should incorporate a defense-in-depth strategy integrating multiple layers of security, such as a zero-trust framework and automation through the use of static analysis tools to help identify potential vulnerabilities in code. It should also include Triple-A defense (Authentication, Authorization, and Accounting), and encryption at rest, in flight, and in use to help ensure data is protected and that only users who are allowed to have access to organizational resources are able to gain access to those resources. Continuous monitoring should also be in place to monitor resources and detect when possible intrusions or unauthorized access has occurred.

**References**

Kueh, T. (2020, January 15). *A Practical Guide to Zero-Trust Security*. Threatpost.

<https://threatpost.com/practical-guide-zero-trust-security/151912/>