**Portfolio Reflection**

Robert A. Rust III

Department of Computer Science, Southern New Hampshire University

CS-405: Secure Coding

Professor Mike Alesso, M.Sc.

March 2, 2025

**Portfolio Reflection**

**Adoption of a Secure Coding Standard, and Not Leaving Security to the End**

Security should not be left as an afterthought when building software. It should be a core part of the software development lifecycle and continuously given attention at every stage of the process. Adopting a secure coding standard such as OWASP secure coding practices or SEI CERT Secure coding standards ensures that the team is on the same page about security practices and that they are followed throughout the software development lifecycle. Starting early with security helps developers recognize vulnerabilities before they become critical issues.

One of the key benefits of a secure coding standard is to identify risks such as SQL injection, denial of service attacks, and buffer overflows. Once developers become aware of these attacks and how they are performed, they can be better equipped to prevent them by coding defensively. Making security a core part of the software development lifecycle from the beginning minimizes costs later.

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

It is not always feasible to have the resources available to practice the maximum level of security for all software systems. It becomes a balancing act between risks and security. Organizations thus need to take context into consideration and ration their valuable resources. Certain organizations will need to have higher security than others, such as organizations subject to data protection laws in fields such as healthcare. Other organizations may not have much sensitive information at all and can have less resources allocated to protecting this information. Risk assessment frameworks such as NIST 800-30 or ISO 27005 provide structured approaches to identifying vulnerabilities and quantifying risks.

**Zero Trust**

The zero trust model switches from a traditional philosophy of "trust, but verify" to "never trust, always verify". This includes entities within the organization's own network, which used to be trusted by default in traditional security models. This is a problem because IT systems have become more and more complex as time goes on, leaving more opportunity for vulnerabilities to sneak in. Once a breach has occurred and gained access into the system, an organization's own entities may be compromised and should not be trusted. Since you never know when a breach has occurred until it is found, it is safer to assume that a breach may have occurred at all times. Some good ways to begin implementing zero trust are enforcing least privilege access, implementing continuous monitoring and authentication, and breaking down a large system into smaller components that each have their own security built in.

**Implementation and Recommendations of Security Policies**

Effective security policies provide a framework for an organization's cybersecurity practices to guide employees to help keep the organization safe. These policies should be clear, enforceable, and continuously improved upon. It is not a one-time process of creating a security policy and never touching it again, it needs to be continuously maintained as gaps and improvements to be made are realized. The team must always be aware of these updates to the security policy, and even if it has not been updated, should be regularly reminded and trained in practicing the policy. The compliance laws and regulations should be closely tracked so that the security policy does not fall behind, causing the organization's compliance to be compromised.