**Module Eight Journal: Portfolio Reflection**

Jeremiah A. Boothe

Department of Technology, Southern New Hampshire University

CS-405-10932-M01 Secure Coding

Dr. Mimi Tam

August 25, 2024

# Introduction

According to Dawson et al. (2010), the earlier a bug is fixed in the Software Development Lifecycle, the lower the cost. A bug fix during the maintenance phase costs between six and seven times more than fixing it during the testing phase, fifteen to sixteen times more than during the implementation phase, and an astounding one hundred times more than addressing it in the design phase. These alarming figures account only for the cost of repairing software errors. However, when software errors lead to vulnerabilities, new risks are introduced that can further escalate costs. The figures do not account for other risks, such as violating privacy laws like [GDPR](https://gdpr-info.eu/), [PIPEDA](https://www.priv.gc.ca/en/privacy-topics/privacy-laws-in-canada/the-personal-information-protection-and-electronic-documents-act-pipeda/), and [CCPA](https://www.oag.ca.gov/privacy/ccpa), which can lead to penalties severe enough to bankrupt organizations. Beyond financial and legal penalties, the damage to trust can be equally, if not more, catastrophic. When trust is broken, users and clients often look elsewhere, and the negative impact can last for years, even decades.

# Adoption of a Secure Coding Standard

Establishing a secure coding standard early in the development process can reduce the likelihood of errors. By integrating security measures from the outset, the chances of introducing vulnerabilities are significantly lowered. Practicing secure coding as a foundational element of the development lifecycle can reduce software errors, that while costly on their own, can lead to even greater problems if exploited by an attacker. There are many standards to pull from, [SEI CERT](https://wiki.sei.cmu.edu/confluence/display/seccode/SEI+CERT+Coding+Standards) for more broad use, [MISRA](https://misra.org.uk/publications/) for embedded technologies serving many different industries, and [AUTOSAR](https://www.autosar.org/standards) for the automotive industry.

# Evaluation and Assessment of Risk and Cost-Benefit of Mitigation

Regular evaluation of potential risks and conducting cost-benefit analyses of mitigation strategies allows development teams to prioritize which security flaws to address based on their potential impact. By doing so, resources can be allocated more effectively, and the most critical vulnerabilities can be resolved first.

# Zero Trust

In a Zero Trust environment, Triple-A security practices are applied, requiring that users, devices, and even internal system processes are authorized, authenticated and logged(accounting). No user or device is trusted by default. The requirement of continuous authorization checks against authentication reduces the risk of unauthorized access, Accounting helps gain insights on attempts at unauthorized access, giving some breathing room to understand motives and intent, and further harden a system.

# Implementation and Recommendations of Security Policies

Security policies should be incorporated into the development lifecycle planning phase and persist through maintenance with the use of monitoring, auditing, and automated detection tools designed to scan systems and check against vulnerability databases.

# References

Dawson, M., Darrell Norman Burrell, Rahim, E., & Brewster, S. (2010). INTEGRATING SOFTWARE ASSURANCE INTO THE SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC). Journal of Information Systems Technology & Planning, 3(6), 49–53. <https://www.researchgate.net/publication/255965523_Integrating_Software_Assurance_into_the_Software_Development_Life_Cycle_SDLC>

Dunham, R. (2018, July 26). *Information Security Policies: Why They Are Important to Your Organization*. Linford & Company LLP. <https://linfordco.com/blog/information-security-policies/>

Kueh, T. (2020, January 15). A Practical Guide to Zero-Trust Security. Threatpost.com. <https://threatpost.com/practical-guide-zero-trust-security/151912/>

Robert, S. (2024, February 14). *SEI CERT C++ Coding Standard - SEI CERT C++ Coding Standard - Confluence*. Wiki.sei.cmu.edu. <https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88046682>

Williams, J. (2024). *OWASP Risk Rating Methodology*. Owasp.org. <https://owasp.org/www-community/OWASP_Risk_Rating_Methodology>