**CS405**

**Final Project: Secure Software Development – PROJ2**

**Project Overview**

This project includes a final presentation video file (Presentation\_FINAL\_Complete23.mp4) located in `mod7/PROJ2, Click "view Raw" for download or use link Project Two: Security Policy Presentation YouTube Video Link: [<https://youtu.be/Kb6ZHVAgG78>]. It covers the critical principles and practices of secure software development, guided by modern cybersecurity standards and risk mitigation strategies.

**Secure Software Development Reflection**

**1. Adoption of a Secure Coding Standard**

Secure software cannot be an afterthought. Adopting a secure coding standard from the outset of development enables teams to prevent vulnerabilities such as buffer overflows, injection flaws, and access control failures. Incorporating standards like [CERT Secure Coding](https://wiki.sei.cmu.edu/confluence/display/seccode) or OWASP guidelines ensures consistent, maintainable, and secure code. When development begins with security in mind, potential threats can be caught early in the software lifecycle, which reduces rework and increases system integrity.

**2. Evaluation and Assessment of Risk & Cost-Benefit of Mitigation**

Risk assessment is a crucial part of developing secure software. It involves identifying and prioritizing risks based on their potential impact and likelihood. Cost-benefit analysis plays a role in deciding which risks are worth mitigating, and which can be monitored or transferred. For example, investing in input validation may cost developer hours up front, but could save millions by avoiding a data breach. The National Institute of Standards and Technology (NIST) provides frameworks that help teams weigh these decisions responsibly ([NIST SP 800-30](https://csrc.nist.gov/publications/detail/sp/800-30/rev-1/final)).

**3. Zero Trust Architecture**

Zero Trust assumes breach. It requires constant verification of users and devices, regardless of their location in or outside of a network. This approach drastically reduces lateral movement during an attack. Implementing Zero Trust principles in software systems includes using strong authentication (e.g., multi-factor), least privilege access, and micro-segmentation. By integrating Zero Trust from the beginning of design, systems become more resilient and adaptive to modern threats.

**4. Implementation & Recommendations of Security Policies**

Security policies guide organizational behavior around software and infrastructure. A strong policy includes clear guidelines on access control, encryption standards, patch management, and incident response. It's critical that these policies are not only documented but actively enforced and updated as threats evolve. Teams should also adopt regular code reviews and automated security testing in their CI/CD pipelines to ensure compliance and reduce vulnerabilities.

**References**

1. Scarfone, K., Grance, T., & Masone, K. (2012). *Guide for Conducting Risk Assessments: NIST Special Publication 800-30 Revision 1*. National Institute of Standards and Technology. <https://csrc.nist.gov/publications/detail/sp/800-30/rev-1/final>
2. The CERT® Division. (n.d.). *CERT Secure Coding Standards*. Software Engineering Institute, Carnegie Mellon University. <https://wiki.sei.cmu.edu/confluence/display/seccode>

**Author**

**Marcus Stanley**  
Robotics Automation & Software Engineering  
CS405 – Secure Software Development