# CS 405 Project Two Script

Brad Mills

Complete this template by replacing the bracketed text with the relevant information.

[**https://youtu.be/OozG3oQtbFs**](https://youtu.be/OozG3oQtbFs)

| **Slide Number** | **Narrative** |
| --- | --- |
| **1** | Welcome to the Green Pace Security Policy Presentation. My name is Brad, and today I will guide you through the essential security policies, principles, and practices aimed at ensuring robust and reliable systems for our development team. |
| **2** | The Green Pace security policy is essential for maintaining a secure software development environment. As our team grows, we need a standardized approach to protect our systems and data. This policy ensures compliance with defense-in-depth best practices, a strategy that employs layered defenses to mitigate risks at every level. |
| **3** | This is our Threats Matrix, where we identify potential vulnerabilities and rank them based on threat levels. For example, unvalidated input ranks as a high threat due to its exploitability, while secure coding issues in logging may rank medium. By using automated tools like static code analysis and dynamic testing, we can proactively detect and mitigate these vulnerabilities. |
| **4** | Our security policy is based on 10 guiding principles:   1. Secure the weakest link 2. Assume vulnerability 3. Use the least privilege principle 4. Employ defense-in-depth 5. Ensure fail-safe defaults 6. Avoid security by obscurity 7. Keep security simple 8. Promote secure defaults 9. Regularly review and improve security 10. Educate developers on secure coding. Each principle is linked to specific coding standards, ensuring alignment and consistency. |
| **5** | The 10 coding standards address key vulnerabilities:   1. Input validation 2. Output encoding 3. Secure logging practices 4. Avoiding hardcoded secrets 5. Data encryption 6. Session management 7. Secure API communication 8. Error handling 9. Dependency management 10. Secure authentication. These standards are prioritized based on risk, with input validation ranked highest due to its widespread impact. |
| **6** | Encryption is critical for securing data. Our policy mandates encryption in three contexts:   1. Data in flight: Use TLS for secure communication channels. 2. Data at rest: Encrypt sensitive data using AES-256. 3. Data in use: Implement secure memory handling to avoid leaks. These measures ensure comprehensive data protection. |
| **7** | Authentication, authorization, and accounting are core to system security. Authentication ensures user identity verification using multi-factor methods. Authorization defines user permissions, employing role-based access control. Accounting tracks user activities to provide an audit trail for accountability and compliance. |
| **8** | We’ll begin with testing input validation. This test ensures that malicious inputs, such as SQL injection attempts, are detected and blocked. For instance, passing a malicious string should result in an error response, while valid input should proceed successfully. |
| **9** | This test checks for secure logging practices. It ensures sensitive data, such as passwords, is never logged. Automated tests confirm that logs only contain non-sensitive information in compliance with our standards. |
| **10** | Our encryption test verifies that sensitive data, such as passwords, is encrypted using AES-256 before storage. The test checks both encryption and decryption processes to ensure data integrity. |
| **11** | Automation is a key component of our DevSecOps pipeline. Security tools like static analyzers are integrated into the build process. Dynamic testing tools analyze the application during runtime in the staging environment, ensuring vulnerabilities are caught at every stage of development. |
| **12** | Addressing vulnerabilities immediately reduces risks but requires significant upfront effort. Delaying fixes increases exposure to potential breaches. For instance, fixing input validation flaws now prevents costly exploits in production. A proactive approach minimizes long-term risks and aligns with best practices. |
| **13** | Our security policy still has gaps, such as insufficient developer training and outdated dependency checks. Future improvements should focus on integrating continuous education programs and automated dependency scanners. Adopting industry standards, such as OWASP’s Top Ten, will enhance our security posture. |
| **14** | In conclusion, Green Pace’s security policy provides a comprehensive framework for secure coding and system architecture. By addressing current gaps and proactively preparing for future threats, we ensure a resilient and secure environment for our team and users alike. |