# CS 405 Project Two Script Template

Jacob Casas

CS-405

02/25/24

<https://www.youtube.com/watch?v=Y5yPNjIyIkE>

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

| **Slide Number** | **Narrative** |
| --- | --- |
| **1** | Security Policy Presentation by Jacob Casas |
| **2** | In this presentation, I will introduce the Green Pace security policy, a comprehensive guide to secure coding practices and system architecture design. This policy aims to:   * Mitigate potential security vulnerabilities in our software. * Promote a defense-in-depth approach for robust security. * Ensure compliance with industry best practices and regulations.   The policy outlines coding standards, encryption strategies, and best practices for authentication, authorization, and accounting. We will also discuss how unit testing and automation can be used to identify and prevent vulnerabilities. |
| **3** | This slide shows the likelihood of different threats affecting our systems. The threats are listed in the table, and their likelihood is indicated by their position in the table. Threats at the top of the table are considered more likely to occur, while threats at the bottom are considered less likely.  The priority assigned to each threat is also listed in the table. Higher priority threats should be addressed first, as they pose a greater risk to our systems. |
| **4** | This slide illustrates our DevSecOps pipeline, which integrates security practices throughout the software development lifecycle. As you can see, security tools and processes are embedded at various stages of the pipeline, ensuring that security is considered from the beginning and potential vulnerabilities are identified and addressed early on. |
| **5** | Same as 4 |
| **6** | This slide highlights our prioritized coding standards, which are essential practices for writing secure and reliable code. These standards address common vulnerabilities and promote secure coding practices throughout the development lifecycle. By following these prioritized coding standards and promoting a culture of secure coding within the development team, we can significantly reduce the risk of security vulnerabilities and build more secure and reliable software products. |
| **7** | This slide outlines our encryption strategy to protect sensitive data at rest, in transit, and in use. Encrypting data helps safeguard its confidentiality and integrity throughout its lifecycle, reducing the risk of unauthorized access or modification. By implementing a comprehensive encryption strategy that addresses data at rest, in transit, and in use, we significantly enhance the security posture of our systems and data. |
| **8** | Here we introduce the concept of Triple-A, which stands for Authentication, Authorization, and Accounting. These three pillars are fundamental to securing access and controlling user activity within our systems. Authentication is the first line of defense, ensuring only authorized users can access our systems. We employ robust authentication mechanisms to verify user identities before granting access. Once authenticated, authorization determines what actions each user can perform. This ensures that users only have access to the functionalities and data they need for their specific roles. Accounting keeps track of user activity, providing valuable insights into system usage and potential security concerns. This information is crucial for maintaining accountability and identifying any unauthorized or suspicious activities. |
| **9** | Unit testing is a software testing technique that focuses on verifying the correctness of individual units of code within a software application. These units can be functions, methods, classes, or modules that perform specific tasks. |
| **10** |  |
| **11** | This slide illustrates our DevSecOps pipeline, which integrates security practices throughout the software development lifecycle. As you can see, security tools and processes are embedded at various stages of the pipeline, ensuring that security is considered from the beginning and potential vulnerabilities are identified and addressed early on. By adopting a DevSecOps approach and integrating security throughout the software development lifecycle, we can deliver secure and reliable software products at a faster pace. |
| **12** | The benefits of taking proactive measures and automating security processes greatly exceeds the drawbacks. Using secure code is the solution. Acting now on security is preferable to having cyberattacks in the future.3 |
| **13** | Some recommendations would include Secure communication channels and Patch management. It is recommended to include a standard for using secure protocols like HTTPS for all network communication within the application and between the application and external systems. This helps prevent eavesdropping and man-in-the-middle attacks. For patch management, this should describe how to find, obtain, and apply security updates for the applications that are installed and utilized in the development environment. Patching vulnerabilities as soon as they are discovered is essential to reducing attacks. |
| **14** | Building upon the identified gaps in the Green Pace security policy, here are additional standards to consider adopting to proactively prevent future security issues   1. Threat Modeling – throughout the application lifecycle, regularly carry out threat modeling studies to find possible threats, vulnerabilities, and attack routes. By taking a proactive stance, security threats are identified and addressed before they become actual assaults. 2. API Security Standards – define and enforce API security standards that address authentication, authorization, data validation, input sanitization, and other critical aspects of API design and implementation. This helps mitigate vulnerabilities specific to APIs and protects against unauthorized access or manipulation of data. |