# CS 405 Project Two Script Template

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

| **Slide Number** | **Narrative** |
| --- | --- |
| **1** | Hello everyone, my name is Brian Chmura, and today I will be presenting the Green Pace Security Policy Guide. This presentation will cover the principles, standards, and best practices our team will employ to ensure that we are protecting our systems from potential security vulnerabilities. |
| **2** | Our security policy is designed to create a unified approach to security within our development team. It was developed to address the increasing complexity of our systems as the team grows and to ensure that everyone remains aligned with our core security principles. This policy is an integral part of our defense-in-depth strategy, which involves layering multiple security controls to protect our assets from various threats. |
| **3** | This threats matrix outlines the key vulnerabilities we have identified within our systems. Each vulnerability has been assessed based on its potential threat level. These range from common issues like input validation errors to more complex risks such as improper authentication mechanisms. Understanding these threats allows us to implement targeted strategies to mitigate risks effectively. |
| **4** | The foundation of our security policy is built on 10 guiding principles. These principles ensure that our coding practices are robust and resilient against potential attacks. Each principle aligns with specific coding standards, helping us maintain a consistent approach across all development activities. |
| **5** | Our coding standards are prioritized based on their potential impact on security. The most critical standards, such as input validation and secure authentication, are placed at the top of the list. This prioritization ensures that the most significant vulnerabilities are addressed first, providing a strong security foundation for our applications |
| **6** | Encryption is a vital component of our security policy. We employ encryption to protect data in flight, at rest, and in use. This includes using SSL/TLS for data transmission, AES for data storage, and encryption of sensitive data during processing. These measures prevent unauthorized access and ensure the confidentiality and integrity of our data. |
| **7** | The Triple-A framework—Authentication, Authorization, and Accounting—is essential for controlling access to our systems. Authentication policies ensure that only verified users can access our resources. Authorization policies define what authenticated users can do, and accounting policies track user actions for auditing purposes. Together, these policies help us maintain tight control over our systems. |
| **8** | Unit testing is crucial for identifying vulnerabilities early in the development process. We use a variety of tests to check for common security flaws, such as buffer overflows, SQL injection, and cross-site scripting. By integrating these tests into our development pipeline, we can catch issues before they make it into production. |
| **9** | Unit Testing Test 1 |
| **10** | Unit testing test 2 |
| **11** | Unit testing test 3 |
| **12** | Security automation is embedded throughout our DevSecOps pipeline. From code scanning at the compilation stage to automated penetration testing during deployment, our tools are designed to catch vulnerabilities at every step. This continuous monitoring ensures that our code remains secure as it moves through the development lifecycle. |
| **13** | In our DevSecOps pipeline, security is integrated into every stage of development. We begin with secure coding practices and use SAST tools during the build phase to catch vulnerabilities in the source code. DAST tools come into play during testing to find issues in the running application. As we deploy, IaC tools ensure secure infrastructure setups. We continuously monitor with SIEM systems and other tools to detect and respond to threats in real time, ensuring our applications remain secure throughout their lifecycle. |
| **14** | When assessing our current security posture, we must weigh the risks and benefits of addressing each issue. Some vulnerabilities pose a higher immediate risk and must be mitigated quickly, while others may be less critical but still require attention. By prioritizing high-risk vulnerabilities, we can protect our systems against the most significant threats while planning for longer-term improvements. |
| **15** | By acting now, we can improve our security posture and respond to threats faster, though it requires initial investment and temporary workflow disruptions. Waiting may free up resources short-term, but increases the risk of vulnerabilities and higher long-term costs due to potential breaches. |
| **16** | Our current strategy lacks proactive security measures and relies heavily on manual processes, creating gaps in response. To improve, we must integrate security into development through the DevSecOps pipeline, increase automation, and enhance collaboration between security, development, and operations teams. |
| **17** | Our current security policy is a solid foundation, but there are areas where we can improve. For example, we need to enhance our threat modeling process to anticipate future risks more effectively |
| **18** | Looking forward, our focus should be on closing these gaps and staying ahead of potential threats. By continuously refining our standards and adopting new best practices, we can ensure that our systems remain secure in the face of evolving challenges. |
| **19** | Finally, here is a list of all the sources referenced throughout this presentation. These materials provide further insights into the security practices we’ve discussed today. |