# CS 405 Project Two Script Template

[**https://youtu.be/ot1SO67FcwY**](https://youtu.be/ot1SO67FcwY) **Project Two: Security Policy Presentation By Martin Richardson 10/20/2024**

| **Slide Number** | **Narrative** |
| --- | --- |
| **1** | "Hello, everyone. Today, I will be presenting on our 'Defense-in-Depth Security Policy,' which focuses on layered security measures to address evolving cyber threats in C/C++ development. My name is Martin Richardson, and I’m excited to walk you through how our policy supports comprehensive security practices through multiple layers of defense." |
| **2** | "This slide provides an overview of our policy. The core principle of defense-in-depth is creating multiple layers of security measures. By implementing firewalls, encryption, and access controls, we ensure that even if one layer is breached, there are additional defenses in place. Our policy also aligns with regulatory standards, ensuring that we comply with industry best practices while safeguarding against various threats." |
| **3** | "Here, we prioritize potential security threats based on their likelihood and impact. We categorize these threats into high, medium, and low priority. High-priority threats, like STD-001-CPP, require immediate attention, while lower-priority threats are still monitored but handled differently. This matrix helps us allocate resources effectively to address the most significant vulnerabilities first." |
| **4** | "Now, let’s discuss the ten principles of secure coding. These are essential for building a strong foundation in our development process. For example, we validate input data, heed compiler warnings, and follow the principle of least privilege. Additionally, adhering to secure coding standards like ‘default deny’ ensures that we mitigate risks at every stage of the development cycle. By following these principles, we reduce the chance of vulnerabilities in our code." |
| **5** | "This slide focuses on specific coding standards we implement to ensure safe and reliable code in C/C++. For instance, we avoid defining C-style variadic functions, ensure memory is allocated correctly, and handle exceptions appropriately. These standards prevent common vulnerabilities such as buffer overflows and memory corruption, which could lead to severe security issues." |
| **6** | "Encryption is key to protecting sensitive data, both at rest and in transit. We implement encryption across three phases: encryption at rest, encryption in flight, and encryption in use. Each phase ensures that data is protected whether it’s stored, transmitted, or actively being used. This multi-layered encryption approach reduces the risk of data breaches and maintains the confidentiality of our information." |
| **7** | "Our security policy also emphasizes the Triple-A framework: Authentication, Authorization, and Accounting. Authentication ensures that users are who they claim to be, through mechanisms like passwords or biometrics. Authorization defines what users can access, ensuring they only interact with data relevant to their role. Finally, accounting tracks all user activities, creating an audit trail for monitoring and forensic purposes." |
| **8** | "Next, we focus on the importance of unit testing in our defense-in-depth strategy. Unit testing is a key part of ensuring that individual components of our code function correctly before they are integrated into the larger system. By testing each part in isolation, we can identify issues early, which leads to more secure and reliable code. We run tests for pop-back functions, erase checks, and out-of-range errors, ensuring that these common vulnerabilities are caught and resolved before they become security risks." |
| **9** | "We leverage automation to make unit testing more efficient. Automation ensures that each time new code is introduced, tests are run automatically to validate that nothing breaks, and all components meet security standards. Tools like Parasoft C/C++test is instrumental in providing automated static analysis and runtime error detection. This continuous testing process helps catch issues as early as possible, preventing costly fixes down the line and contributing to overall system security. |
| **10** | "Automation plays a crucial role in enforcing security compliance. In the production phase, we automate tasks such as penetration testing to ensure our systems are secure before deployment. Tools like Parasoft C/C++test and Intruder allow us to automate vulnerability scanning and testing without sacrificing speed. By integrating security into the development pipeline, we maintain high standards of security without hindering productivity." |
| **11** | "Here, we weigh the pros and cons of our defense-in-depth strategy. On the positive side, our policy ensures comprehensive security coverage, adherence to best practices, and compliance with regulations. However, the complexity and development overhead can be challenging, potentially leading to higher costs and resistance from teams. It’s important to balance the benefits of security with the practicalities of implementation." |
| **12** | "Lastly, I’d like to offer some recommendations. First, our policy would benefit from a comprehensive Incident Response Plan to ensure we’re prepared to handle security breaches effectively. We also need a stronger focus on data privacy controls, especially considering regulations like GDPR. Lastly, while automation is helpful, we shouldn’t rely solely on it—regular human oversight is critical for catching complex threats." |
| **13** | "To conclude, by following these defense-in-depth strategies, we can ensure our systems are protected against a wide array of threats. From secure coding principles to encryption and automation, each layer adds an essential component to our overall security framework. Thank you for your attention, and I’m happy to answer any questions you may have." |