# CS 405 Project Two Script Template

**Andrew Black, October 14th, 2022, Project Two: Security Policy Presentation**

<https://youtu.be/5NTYG83hMAs>

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

| **Slide Number** | **Narrative** |
| --- | --- |
| **1** | Hey everyone! In this presentation, I will be demonstrating some security policy recommendations based on the guidelines and policies I have set forth with my document. Let’s get started. |
| **2** | Defense in Depth is the process of layering security methods and practices to ensure a more secure system. This security policy is to provide a set of guidelines for developers to write simple, clean, and secure solutions. |
| **3** | Within the security policy is a list of threat likelihoods and prioritizations. This threat matrix describes it simply: “likely” means that threats are likely to happen, while “unlikely” means that threats are not as likely to happen. Likewise, “priority” means the standard has priority while “low priority” means exactly that. |
| **4** | Outlined in the security policy are ten principles that should be followed when securely coding. These principles were obtained from the Tricksonflicks blog, a security-focused blog. In essence, we should validate all input, heed any and all compiler warnings where necessary, architect and design to comply with security policies, keep it simple, use default-deny practices, adhere to the principle of least privilege, sanitizer data sent to other systems, practice defense in depth, use effective quality assurance techniques, and adopt a secure coding standard. |
| **5** | There are also a set number of coding standards listed that are specific to C and C++. Starting from the top: we should not cast to an out-of-range enumeration value, we should guarantee that container indices and iterators are within the valid range, we should NOT attempt to create a std::string from a null pointer, we should NOT access freed memory, we should detect and handle memory allocation errors, we should use a static assertion to test the value of a constant expression, we should NOT rely on indeterminate values of errno, we should NOT form or use out-of-bounds pointers or array subscripts, we should cann only asynchronous-safe functions with signal handlers, and we should clean up thread-specific storage. |
| **6** | Moving on, there are several well-known encryption policies based on the different types of motion: encryption at rest, encryption at flight, and encryption in use. Encryption at rest is a form of encryption where the encrypted data is at rest, such as in a database or cold storage. Encryption in flight is a form of encryption where the encrypted data is actively being used, such as in an HTTPS request between the server and client. Finally, encryption in use is where encryption is applied at every stage of the process, never becoming insecure. This is less common. |
| **7** | The triple-A policies are authentication, authorization, and accounting. They are simple to follow along with: authentication is where the identity of the user is proven, authorization is the process of granting the authenticated person access to the systems they only need access for, and accounting is the process of auditing and logging authentication and authorization to ensure policies are being applied correctly. |
| **8** | Unit testing plays a huge role in the software policy as without it, edge cases and unexpected bugs may come up when they are least expected. I have tested numerous code vulnerabilities on collections as a part of a prior assignment, both positive and negative. |
| **9** | Automation is a process that never ends as it pertains to system design and software development, but integrating DevSecOps is crucial for the process to be secure all the way through. |
| **10** | The DevSecOps pipeline is akin to the DevOps pipeline but with extra attention paid to security through the use of various tools. Some external tools include the use of compiler warnings and automated secure code testing tools such as CodeSonar, Helix QAC, Astrée (astray), and more. |
| **11** | With the implementation of the security policy comes risks of not following it, which are outweighed immensely by the benefits *of* following it. Determining whether to act now or to wait can be a no-brainer in many security-related situations. I would like to believe that in most cases, people do not intentionally wait to implement security solutions, but rather, miss vital, subtle holes in the code that they write which can lead to abuse later. One way to help counteract this is with the use of static code analysis tools detailed in the previous slide. If the use of these tools is delayed, it risks anyone with a keen eye of exploratory mood to find and abuse an unseen vulnerability in the code. The implementation of a DevSecOps cycle is critical to the success and safety of a codebase, and further, the company. |
| **12** | Staying on top of the latest security trends is the most critical recommendation; computer and software security is a never-ending battle between attackers and defenders that reveal new pieces of information every so often. For example, a Java package by the name “Log4J” had a security vulnerability that allowed attackers to perform remote code execution through a vulnerable protocol. Given how large-scale the package is and how it is used in many software applications around the world, it is critical to stay on top of trends like these. |
| **13** | All in all, the security guidelines and principles outlined within this document gives insight into the world of cybersecurity and secure coding, and an ideal explanation of actions that must be taken to ensure that Green Pace writes safe and secure code. Ensuring the safety of the code written is vital for the success of the company, as well as implementing other security policies outside of that, such as multi-factor authentication and a zero-trust policy to ensure that only employees have access to what they only need. |
| **14** | I hope you found my presentation enjoyable and informative in the world of secure coding, and thanks for watching! |