# CS 405 Project Two Script Template

Joshua Eberharter

9 December 2022

Project Two: Security Policy Presentation

https://youtu.be/L0erlfW\_jC8

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

| **Slide Number** | **Narrative** |
| --- | --- |
| **1** | Hi, my name is Joshua Eberharter. I’ll be sharing our security policy here at Green Pace for our software development team. |
| **2** | The first thing I’d like to talk about is defense in depth. The whole idea of DiD is to establish several barriers across multiple layers of the system to protect valuable data. This, in turn, will prevent vulnerabilities in code and push developers to use best practices. |
| **3** | The threat matrix prioritizes threat levels using priority levels, remediation costs, likelihood, and severity levels. Higher priority threats tend to be more severe, while low severity threats typically have short workarounds or lesser consequences. |
| **4** | These are our ten principles and their associated standards. We have validate input data, heed compiler warnings, architect and design for security policies, keep it simple, default deny, adhere to the principle of least privilege, sanitize data sent to other systems, practice defense in depth, use effective quality assurance techniques, and final adopt a secure coding standard. |
| **5** | For our coding standards, they are listed in order of priority. We have SQL injection, memory protection, string correctness, input/output, integers, data type, data value, expressions, exceptions, and assertions. Since both SQL injection and memory protection can jeopardize the sensitive data of others, they have been placed at the top. Three through eight concern data manipulation and staying within best practices. The final two, exceptions and assertions, ensure code quality is prioritized and code conditions are properly tested. |
| **6** | Encryption at rest protects data that is stored. This could be more traditional storage medium like mechanical or flash drives, or a more modern medium like cloud storage. Encryption in flight protects data that is being transmitted from one place to another, typically over the web. This data is protected using security protocols like SSL/TLS/HTTPS. Finally, encryption in use protects data that is being accessed, processed, or viewed. This stage of encryption occurs in between rest and flight and is enforced using memory protection techniques like encryption keys. |
| **7** | Authentication verifies the identity of the user. This can be accomplished by using techniques like a basic username-password combination, biometrics, and two-step verification. Authorization, on the other hand, determines the privileges of the user. It’s important for users to have the least amount of privilege required to do their job. This ensures users cannot access information that is not intended for them. Accounting is used to log information in records to keep track of common transactions and event changes. Accounting can be used to ensure authentication and authorization policies are working properly. |
| **8** | Automated unit testing can benefit developers by eliminating the need to write manual tests and consequently saves times. For example, verifying basic functionality for a container library. Individual tests can be created for each method to reduce potential errors and build more robust code. |
| **9** | This is our DevSecOps pipeline. It is broken up into pre-production phases and production phases. Pre-production phases start with assess and plan, design, build, and verify and test. Once in production, we more to transition and health check, monitor and detect, respond, and maintain and stabilize. |
| **10** | The DevSecOps pipeline incorporates security policies into our software development lifecycle. In the pre-production phases, we added security tool training and in-depth threat analysis, additional security measures within the IDE, and static and dynamic testing techniques, like unit, system, and integration testing. In the production phases, monitoring is emphasized, and we added more metrics logging to adjust and eliminate bottlenecks, penetration testing, and more feedback from end users. |
| **11** | For risks and benefits, it’s important to prioritize bugs that have higher severity. Waiting to fix these defects can cause several more bugs, longer development times, unsatisfied customers, costly fixes, and, overall, a non-secure system. To prevent this, we conduct security audits and discuss security needs in daily team meetings. |
| **12** | There are gaps in this security policy. For example, there is no clear procedure for backing up data and keeping it secure. There are also limited policies concerning direct access of data, and authentication and authorization roles. Additionally, there are no response plans in case of a data breach. These are all things we would like to see in upcoming iterations. |
| **13** | To conclude, our security policy is to be followed by new and old developers alike. Developers can use these standards and principles to build secure, robust code that will last. Security is a mindset that needs to be adopted from the beginning of development. Security cannot wait until the end. |
| **14** | Thank you for listening. |