# CS 405 Project Two Script Template

Name: Tim Gallus

Date: 2/19/2023

Assignment: Project Two: Security Policy Presentation

YouTube link: <https://youtu.be/YqBqEKJj1YI>

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

| **Slide Number** | **Narrative** |
| --- | --- |
| **1** | Hello everyone. My name is Tim Gallus and this is my Green Pace Security Policy Presentation. |
| **2** | We will begin with an overview of defense in depth. A defense in depth approach to security provides a layered defense, forcing attackers to circumvent multiple layers of security and mitigation strategies. Part of this strategy includes looking at security in the form of authentication, authorization, and accounting. Although not strictly part of defense in depth, we also need to be sure that we are proactive in designing with security in mind rather than pursuing it as an afterthought or during implementation. |
| **3** | Our threat matrix in slide 3 shows a number of the vulnerability causes associated with our coding standards grouped under if they are likely or unlikely, and whether they are a priority or not. It’s worth noting that the likelihood of a threat occurring is not the only factor that goes into deciding if a threat is a priority to secure against. |
| **4** | This next slide goes over the security principles for our security policy. As you can see in the slide it also shows some of the associated coding standards that are applicable to each principle. Our 10 principles are: 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 adopt a secure coding standard. |
| **5** | We can see here a list of our 10 coding standards, organized by their priority. As we mentioned in a previous slide, a number of factors go into determining how these are prioritized including not just the severity of the threat but also the likelihood and remediation cost, all of which are shown in this chart. Our coding standards are: Sanitize data passed to complex subsystems, Ensure that unsigned integer operations do not wrap, Guarantee exception safety, Value-returning functions must return a value from all exit paths, Free dynamically allocated memory when no longer needed, Do not modify the standard namespaces, Understand the termination behavior of assert() and abort(),Do not attempt to modify string literals, Predicate function objects should not be mutable, and Implement abstract data types using opaque types |
| **6** | Encryption is an important part of security and a defense in depth policy. There are 3 types of encryption: Encryption at rest covers the encryption of stored data that is not in active use, Encryption in flight is the encryption of data and communications as they are actively being transmitted, and finally Encryption in use is the encryption of data that is actively being utilized. This slide goes into additional detail regarding these three area of encryption. |
| **7** | Triple-A policies involve authentication, identifying users and verifying that they are legitimately allowed access to the system, authorization, which is deciding what permissions a legitimate user should have, and accounting, which is tracking actions of a user on the system. Authentication involves such things as multifactor authentication and strong password policies, authorization involves things like default denial and granting least necessary permission and only while needed. Accounting is of help in discovering and mitigating damage done. |
| **8** | The next few slides are examples of some of our available unit tests, including a screenshot of them running and returning results so you can see what that looks like. Unit tests are a form of automated testing that allows us to test small pieces of code early and often. It is very helpful for both security and functionality testing. |
| **9** | This slide shows us an easy to follow diagram of our dev sec ops pipeline. We’ll be discussing our use of automation as a part of it in the next slide. |
| **10** | Automated enforcement of the standards encompassed by this policy should happen at code check in, at the interstitial line between design and build. Whenever new code is checked in to a separate branch before it merges to main, it should trigger automated tests to run the various tools listed in the various coding standards that have been set up as part of the check in pipeline for this purpose and raise awareness of any security policy violations. Unit testing should also occur at this point. Code that does not meet the security policy requirements should fail check in compliance until it is fixed. There should also be an automated request triggered on code check in to request a peer review.    Additional automated testing should be included as part of the Verify and Test phase to retest the full build rather than focusing on newly checked in code. Some automated collecting of logs and detection can be added to the Monitor and Detection to watch for violations of these security policies occurring or being exploited. |
| **11** | Implementing a defense in depth security strategy proactively based on a uniform and reliable security policy can drastically reduce vulnerability to security breaches and costly mitigation and retrofitting of code design and implementation to secure vulnerabilities after the fact. The primary costs of doing so are additional time to plan and design with security and defense in depth in mind from the start and the additional training to make sure everyone at the company is aware and practicing the chosen security policy and other good security practices. However, waiting until after during or after code implementation to implement security, or not having a unified security policy can lead to costly code redesigns during or after implementation along with unexpected vulnerabilities being left in place. Both of which can be much more costly in the long run. |
| **12** | The current security policy is an initial base that has areas where it could use additional improvement and elaboration. These include:   * User security training. Poor password security, unattended work devices that may be accessed by malicious parties, and vulnerability of the human element to social engineering attacks is one of the more difficult vulnerabilities to fully secure. Our policies did not cover this strongly, but the company should put strong policies and training in place for everyone with regards to these. * Penetration and other “white hat hacker” security testing for security vulnerabilities. Regular and knowledgeable testing may find allow security vulnerabilities to be closed before they come to the awareness of malicious parties. * Nothing in our policies discussed physical security of hardware assets or the need to use reputable cloud assets that are suitably secure. * Our current security policies do not go in depth about the steps we will take to provide accounting beyond going into its importance. |
| **13** | It is important moving forward to keep the tools and software that we use up to date, to remain aware of new and existing security threats, to be proactive in our security planning rather than seeing security as an afterthought to development. Educating ourselves and others on the security policies of the company and making sure we select security policies that are best for our industry is important and just like security practices, security threats are always evolving. While constant change to our policies is not helpful, we should review our security policies at appropriate intervals or when new threats develop and adjust them to meet the changing needs.  As individuals we should stay aware of the security policies of this company and general best security practices and actively make use of them. |
| **14** |  |