# CS 405 Project Two Script Template

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

| **Slide Number** | **Narrative** |
| --- | --- |
| **1** | This is a presentation on the security policy developed for Green Pace. |
| **2** | Defense in Depth presents the idea of incorporating multiple layers of security through the software development lifecycle to ensure they overlap and provide coverage. This Security policy aims to provide guidelines for each of the layers seen in this diagram to ensure sufficient security. |
| **3** | A list of the security standards and their respective priories can be seen here. |
| **4** | Ten coding principles were established, and you can find the security standard they map to listed for each. |
| **T** | The remaining five principles and their respective security standards can be seen here. |
| **6** | A more detailed explanation of the coding standards can be seen, including SQL Injection, String Correctness, Memory Protection, Data Value, and Container Indices. |
| **7** | The remaining coding standards can be seen here, which include Object’s Memory Lifetime, Exceptions, Data Type, Reserved Identifiers, and Assertions |
| **8** | There are three primary forms of encryption for data. Data can be at rest, in transit, and in use. A thorough description of each form and how to implement commonly used forms of each ecryption can be seen here. |
| **9** | Triple A Policies aim to incorporate Accounting, Authorization, and Authentication into a system. |
| **10** | Unit Testing can be a solid form of security measures that ensure code works as expected. A series of Unit Tests were performed to demonstrate this. |
| **11** | This Unit test is verifying the erasure of a collection. This uses the ASSERT\_TRUE assertion to confirm the collection is properly emptied after using the erase method. This demonstrate Properly Deallocating Memory. |
| **12** | This Unit test is verifying that the push\_back method actually moves the newly added value to the end of the vector object. This demonstrate data type, data value, and quality assurance. |
| **13** | This Unit Test is verifying that an error is thrown when attempting to resize a vector to a negative value. This falls under the security vulnerability for string correctness and data value. |
| **14** | This Negative Unit Test is verifying an exception is thrown when attempting to access a vector at an index outside of what actually exists. This falls under memory proper use of assertions and checking for String Correctness. |
| **15** | DevSecOps is the practice of incorporating Security throughout the CI/CD pipeline and DevOps practices. Automation can be incorporation into several stages of this. |
| **16** | The DevSecOps Pipeline is used to illustrate the key stages of the Software Development Lifecycle that security can be incorporated into. This approach incorporates security through the Continuous Improvement Continuous Delivery Pipeline. The key stages include: Assess and Plan, Design, Build, Verify and Test , Transition and Health Check, Monitor and Detect, Respond, Maintain and Stabilize |
| **17** | Examples of tools that can be used for this include: Security Tools can be implemented in essentially any stage shown in the diagram. To demonstrate, here are a few examples:  Planning and Design: Threat modeling tools such as Threat Dragon from OWASP.  Build: Automated IDE plug-ins and compiler warnings.  Verify and Test: Static Testing tools such as OWASP ZAP, Parasoft, or JBroFuzz  Transition: Tripwire, Chaos Monkey tool, Principle of Least Privilege, and configuration management tools such as Puppet and Docker.  Monitor and Detect: Runtime application self-protection (RASP) |
| **18** | Regarding the risks and benefits, problems and solutions of deciding to wait or act now, our security strategy is designed to address the problems of security vulnerabilities in code and systems, lack of automation in security testing and deployment, and insufficient training and resources for secure coding practices. Risks and Problems of waiting include damage to reputation and trust, failure to comply with regulatory standards. Solutions and benefits of the policy include implementing security into the DevOps and CI/CD pipstandardeline, implementing automation, and strong education of the security |
| **19** | The security policy as a whole lacks substance, cohesiveness, clear guidelines, and training details to aide implementation and adoption. The Code Standards are too vague, the Code Principles are vague as well without any direct correlation to the company’s specific and unique needs, and little to no information has been provided as to what the client needs, currently has, and general context. There is no Incident Response Plan or specific and clear guidelines that extend beyond standard coding practices. |
| **20** | To address the severe lack of relevant and and specific coding standards, security frameworks that are use widely and are trusted should be implemented and followed, such as OWASP or NIST frameworks |
| **21** | It’s crucial for NIST and OWASP frameworks to be implemented throughout the security policies at some point to ensure any future gaps are promptly addressed. It’s also recommended that an Incident Response Plan be developed |