# CS 405 Project Two Script Template

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

| **Slide Number** | **Narrative** |
| --- | --- |
| **1** | Hello, my name is Dustin Haugh and I would like to introduce our Security Policy for Green Pace. |
| **2** | Defense in Depth is a strategy to create protective layers of security into our application. One of those layers is our approach to secure coding this policy will cover. |
| **3** | The following threat matrix covers the probability and severity of each secure coding policy as described by SEI CERT C and C++ definitions. They range from Likely to Unlikely and Low Harm to High Harm. As you can see in this threat matrix, many of our policies cover coding threats that are both Likely and High Harm. |
| **4** | Here we can see how the various coding standards in our policy meet our ten principles. |
| **5** | We can rank our coding standards by importance when considering both cost to repair and how likely this error might occur. We have L1 which is the highest level of likelihood, severity, and cost to repair and L3 which is least likely, least severe, and least expensive to repair. |
| **6** | Data can be at rest, on the move, or in use. We have developed policies to encrypt data at rest using AES encryption, data at flight with AES keys, and encryption and decryption methods with data in use. |
| **7** | Triple-A policies are used to authenticate a user, restrict their access to only what they are authorized, and account for their actions. This allows us to add another layer of security to our application and perhaps find suspicious events. |
| **8 through 11** | Unit Testing is a good way to quickly troubleshoot errors in an application by using an Assert() macro. In this case we are using GTest (Google Unit Testing).  Slide 9:  This is example of MaxSizeIsGreatorOrEqual checks for the number of objects in a collection.  Slide 10:  CapacityIsGreatorOrEqual tests to verify the capacity of a collection with different methods.  Slide 11:  These methods are examples of how we can verify resizing a collection of objects. |
| **12** | DevSecOps Diagram: Explain where the security tools reside in the flow of automation. State which stages will contain security automation. For instance, when will the compiler be used?  Within the DevSecOps Diagram, security begins as a consideration at the beginning of planning and design.  During code development code analysis can be performed with automated tools like CppCheck and our compiler.  We can also look for errors by using unit testing. |
| **13** | As mentioned before, one of the tools we used to check for code vulnerabilities is CppCheck. These types of tools can give us a more thorough idea of code behavior concerns than what our compiler might. |
| **14** | There is a great benefit to thinking about security from the beginning of the software development cycle, including overall cost and liability. |
| **15** | Security is an ongoing process. Threats change and evolve, and so does the support for the languages we code with. It can be important to look for changes in secure coding policies annually, and possibly talk to white hat hacking sources to stay up to date on new threats. |
| **16** | Adopting best practices and secure coding standards are fundamental to software development. Utilizing Defense in Depth strategy and taking security into consideration from the beginning of development is a strong strategy for successfully safeguarding Green Pace. |
| **17** | Thank you so much for listening to this presentation! |