# 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 Nicholas Cleveland and this is my policy guide for Green Pace to provide implementation guidelines and recommendations for maintaining it in the future. |
| **2** | The overall goal of this policy is to have security standards and policies that are applied in daily work. This will help to stop attacks that arise and assumptions of vulnerability. |
| **3** | These are the list of the threats that are possible based on the security policy and standards. |
| **4** | These are the 10 core coding standards to follow to ensure secure coding.  Validate all input from all sources of data that are untrusted. By properly validating input data you can help and eliminate software vulnerabilities.  Code should be compiled by using the top warning accessible for the compiler and eliminate warnings.  Security policies should be enforced by creating a software architecture and design that implements this.  The overall design and layout should be small and simple. More complex designs could cause more errors.  Default access to the software should be denied. This helps to protect access to the software and only allows access when it is permitted.  There should be a set of privileges that execute when completing a task. This helps to minimize the time an attacker has when gaining access.  Data should be sanitized when passed to other complex subsystems to help with controlling attackers when they invoke unused components.  Have multiple defense tactics to help and stop and attack. This can help and make it so that if an attacker makes it through one layer the next one is there to stop them from continuing.  Eliminating and identifying vulnerabilities can be accomplished by having good quality assurance. By having good testing, it can lead to a more secure system overall.  By having a coding standard that is secure you can help and maintain a safe environment for your target development language. |
| **5** | Data Type - Do not define a C-style variadic function.  Data Value - Do not cast to an out-of-range enumeration value.  String Correctness - Guarantee that storage for strings has sufficient space for character data and the null terminator.  SQL Injection - Prevent SQL injection.  Memory Protection - Do not access freed memory.  Assertion - Expressions used in assertions must not produce side effects.  Exceptions – Handle all exceptions.  Input/Output - Do not alternately input and output from a file stream without an intervening positioning call.  Object Oriented Programming - Do not invoke virtual functions from constructors or destructors.  Platform Security - Do not allow privileged blocks to leak sensitive information across a trust boundary. |
| **6** | Encryption in rest involves stopping an attacker from gaining access to unencrypted files by ensuring the data is encrypted when on disk. Examples of this are the Payment Card Industry Data Security Standard (PCI) or the Health Insurance Portability and Accountability Act (HIPAA)  Encryption at flight involves stopping an attacker from gaining access to data that is encrypted when it moves over a network. Example of this is Transport Layer Security(TLS)  Encryption in use involves stopping an attacker from gaining access to sensitive data that is being used. That data is never left unsecured regardless of its status. |
| **7** | Authentication is the first goal in providing a secure system for identifying users who have access to your application. The user needs to be able to authenticate by making sure that the user is who they say they are. Some examples of authentication are things such as security questions and 2 factor authentication.  Authorization involves providing the type of access the user will have within the application. This helps to make sure that the right people are accessing certain aspects of the software. An example of this is having role-based access control to help secure the application. Limiting access to sensitive areas helps to make it more secure.  Accounting is important in helping to monitor what the user is doing once they have passed through the other parts of the framework. You should know what each user is accessing or attempting to access to help to know if changes need to be made. It can also help to see if the user was authenticated correctly. |
| **8** | [Insert text.] |
| **9** | Automation will be used for the enforcement of and compliance with the standards defined in this policy. Green Pace already has a well-established DevOps process and infrastructure. Define guidance on where and how to modify the existing DevOps process to automate enforcement of the standards in this policy. Use the DevSecOps diagram and provide an explanation using that diagram as context.  The first thing we will do in pre-production is assess the threat landscape, changes, and provide a backlog of those changes. We will respond to those threats as needed. We will then design a test-driven practice to help with the security of the software. Next, we can build a secure space for the automation to be executed. To end preproduction, we will verify everything we have implemented and test for vulnerability, functionality, compliance, and security. Full production will take place next. It will start with configuring and deploying security settings. Then we can monitor everything and make sure to log all the data being accessed. We will then respond to threats and block any attacks by turning off necessary services. Finally, we will maintain and stabilize the system. |
| **10** | The core of a DevOps pipeline constitutes the following: continuous integration/continuous delivery (CI/CD), continuous testing (CT), continuous deployment, continuous monitoring, continuous feedback, and continuous operations. |