# 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 Daoud Sogoba and this is my presentation on the security policy of the Green Corp Corporation. |
| **2** | The policy found within relies on a key piece of security policy. It is defense in depth. Security apparatuses are most effective when used in the defense in depth strategy. Defense in depth being a redundant multilayered defense. In software development this ranges from the use of input verification at the code level to the physical barriers and security guards found in the data storage facilities of cloud service companies. |
| **3** | My threats matrix shows that though most of the standards I have chosen to focus on are not the most likely to occur, I still find them still be of high pretty in preventing. Unlikely issues can still be valuable vectors for threat actors.  The threat that are the most likely and of the highest priority are “Ensure adequate space is present for string character data.” and “Use prepared statements for SQL queries”. Adequate space for string character characters removes a vector for buffer overflow, while the use of prepared statements prevents threat actors from injecting malicious code via SQL statements.  Though still just as likely as the former material, the standard “Do not attempt to access a variable after its lifetime” is still a high priority. This is an error that will prevent the software from compiling as references to objects that no longer exist is not possible.  While this list only contains standards that are considered unlikely. I do still consider them a higher than low priority but not a higher priority than standards in the likely category. This is to do my philosophy that as many standards as possible should be kept in compliance regardless of their likelihood. |
| **4** | The 10 principles for secure coding in this policy are {READ POLICIES OF SLIDE}. I have the coding standards mentioned in the previous slide aligned with the principles that best match them.  Validating user input is imperative as it prevents errors that are a result of unexpected user input. This includes buffer overflow.  Compiler warnings offer developers insight into issues that will not prevent the code from compiling but that may present security risks.  Software should be designed from the ground up and the starting points to incorporate security best practices to ensure a secure product.  Simplicity is a boon to developers and customers alike. It allows developers the ability to modify and secure their code and protects consumer data more easily.  Default deny ensures that developers have not possibly way to accidently forget to remove permissions that users or thread do not need.  The principle of least privilege ensures that objects and users are not given more permissions than their tasks require. This removes potential attack vectors.  Sanitation of data sent to other systems ensures that data that is not pertinent to the performance of a task by another system is kept separately.  Defense in depth ensure that are numerous and redundant systems to counter and protect against as many threat vectors as possible.  Effective quality assurance like unit testing can help ensure that software is secure and almost always operates within the bounds of expected behavior.  Adopting a secure coding standard ensures that the software written will have a focus on security from the moment it starts being worked on. |
| **5** | The coding standards form the previous slides are presented here together to display their varying levels of severity, likelihood, remediation cost, priority, and level. This again was provided by the Carnegie Mellon University. |
| **6** | Encryption at Rest: Data at rest is one of the most secure states available. Disk encryption allows sensitive data to be secured even if the physical media storing the data is compromised or stolen  Encryption at flight: This refers to data like encrypted emails making use of PKI or SMIME to ensure that data remains encrypted as it is moved across a network.  Encryption in use: This data currently being accessed in memory and is arguably one of the more vulnerable states of data. Here data is decrypted so that it can be used to perform the actions required of the computer. Some systems make use of Memory Encryption in system or CPU ram. |
| **7** | * Authentication - Ensures that users are who they say they are to gain access to a system. Used for positive identification * Authorization - Grants or Denys users the authorization to make use of a system * Accounting - Logs will be kept on system authorizations, authentications, and accesses. |
| **8** | A type of testing in which functions or units of whole pieces of software are tested. |
| **9** | * In this test I am testing the assertion that the resizing of the collection causes a decrease in the collection size. |
| **10** | * In this test I am testing the assertion adding 10 entries to the vector collection does in fact increase the size to 10 entries. |
| **11** | * In this test I am testing the assertion that the resizing of the collection outside of its limits throws a range exception error. |
| **12** | * In this test I am testing the assertion that the reservation of entries space increases the size of the collection. |
| **13** | Automation in the DevSecOps pipeline is a natural part of the implementation of this security policy. There are several tools that allows developers to test their software and code for varying vulnerabilities. |
| **14** | Development security operations is a process by which developers continuously use and integrate security policies in tools during software development. This is often accompanied by several checks for security breaches or potential attack vectors before and after code is written and while it is running.  These tools include but are not limited to Fortify which tests against HTTP response splitting and Code Sonar, a tool that tests against sql injection. |
| **15** | When deciding whether to implement security policies at the start or at the end of a project, developers often find themselves struggling to find a balance with implementation timelines, project funding, and software completion.  Some of the primary benefits of incorporating security into the software development lifecycle early on are secure coding policies and techniques will be core to the software large, security vectors will be identified and fixed before software released, and a more secure product can be expected. Granted these often come with an increase in cost and project complexity, as well as an increased development time.  There are few if any pros too waiting to implement security policies in the software development lifecycle. the primary being a fast development time at the cost of security policies potentially never being implemented. Whether that's due to a lack of time or project funding. Or what is likely more common, halfhearted, or incomplete attempts at implementing security policies and standards in the software. |
| **16** | For the green pace corporation, there are several recommendations to ensure proper security policies for software development. Use include or not limited encryptions for all three states of data, the implementation of the AAA framework, and the insurance of developer knowledge and education regarding implementation of software security and security coding policies. |
| **17** | To conclude, it's great that the policies focusing on security and security coding are likely to prevent many, if not most of the common threat actors and attack vectors that software often finds. But it is imperative that we stay vigilant to the ever-present and ever evolving threats. Any cyber security spaces. Continual growth of knowledge bases and defense in depth of implementation will ensure that software and company data remain safe and secure. |