# CS 405 Project Two Script Template

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

| **Slide Number** | **Narrative** |
| --- | --- |
| **1** | Hi my name is Nicole Penner. Welcome to my presentation on the security policy for Green Pace. |
| **2** | This security policy is needed to help make sure the system is as secure as possible. This policy will support defense in depth by helping to create more layers of security. If one level of security fails, then the next in place will pick up where it left off. |
| **3** | This table shows how likely a threat is to happen and compares it with level of priority it takes. The sections are divided into likely with low priority, unlikely with low priority, likely with high priority, and unlikely with low priority. To decide how these are divided, it is important to look at the cost of fixing the issue beyond its likelihood to happen or not. If an event is likely to happen and has a high priority that is because it has the potential to cause a lot of damage and will be expensive to fix. Priority level of the unlikely events is also based on this. This makes it the easiest way to divide the policies. |
| **4** | The 10 principles of security are: validate user input, heed compiler warnings, architect and design for security policies, keep it simple, default deny, adhere to the principles of least privilege, sanitize data sent to other systems, practice defense in depth, use effective quality assurance techniques, and adopt a secure coding standard. The coding standards that apply to each principle are included as well. |
| **5** | My coding standards are not placed in any specific order. I did this because I believe they all are important, and are important in different ways. If I had to choose a method of prioritization it would be based on the threat’s matrix. If a threat is more likely to occur and is more expensive to fix it would be more important to deal with that standard first and foremost. If it is about the same for multiple standards, then they are treated equally and so on. |
| **6** | Encryption At rest: This refers to the initial state of digital data, when it is at rest means it is stable and not traveling through the system or being acted on.  Encryption at rest depends on what physical or logical data sources or storage need to be protected. Data should be encrypted prior to saving to the disk to protect it. This could be done with MySQLServer or other tools.  In fight: This refers to data when it is “in motion”, as in being transmitted.  Data should be sent through secure and encrypted channels, such as by HTTPS. Data should be encrypted before it is sent, and decrypted after the intended person receives it so that it can’t be hacked and altered while in flight.  In use: This refers to protecting data that is not just passively being stored. This could include data that is being edited, accessed, or processed by the system.  Data should be handled as securely as possible while in use, this includes never storing a password unencrypted. |
| **7** | Authentication: The first step in the Triple-A Framework process, this is how the system or network verifies a user’s identity. It makes sure the user is who they are claiming to be. This is checked typically using login credentials like username and password, biometrics, two-factor authentication, and one time use passwords.  Authorization: This is the second step in the Triple-A Framework process, it checks what the user is allowed to access in the system after they have been authenticated. This could be what they have access to or what they have the ability to make changes to. This helps to keep the system safe from anyone who shouldn’t have access to certain things, or shouldn’t be able to make changes to the system.  Accounting: This is the third step; it allows for the events or actions a user performs while they access the system to be logged. This helps to ensure that the system is secure, and if there is an issue it may be able to be pinpointed to one user. A user logging in would be accounted for, or even just attempts to login. It is very useful to helping to maintain secure standards of a system. |
| **8** | This test is a positive one, it checks to see if the collection is empty at the start of the test. The collection should be initialized with correctly, and have nothing in it. This goes with they idea of STD\_010\_CLG. The bounds should be defined of an array… though this test does not check that maybe one should be added that does or it should be included since it is a coding standard for this document. |
| **9** | This unit test checks to see if the reserve size increases, but that the size of the collection does not. Here is the code for the test, which is a positive one. The coding standard this follows is STD-005-CPP where memory should not be accessed once it is freed. The other standard is STD-006-CLG where static assertions should be used to test code. |
| **10** | This test checks to see if the out\_of\_range exception is actually thrown when an index that is called that is out of bounds of the array. This follows standard 5, do not access memory that has been freed. This also follows standard 6 where assertions should be used to test the code. Standard 7 is also followed because this test shows an exception is handled properly. I don’t believe this test needs to be more in depth, it does a good job. |
| **11** | This is a negative unit test. It checks that if we remove an entry, we check if there is an entry there. It will show that there is not one there, therefore verifying this unit test is correct. This follows the coding standard, which is number 6. That static assertions should be used to check and test the code. There are no changes I would make to this test. |
| **12** | This just shows that all the unit tests, beyond the specific ones I discussed were successful when they ran. This was the result, if they were not successful, we would see red instead of green on the specific test. Unit tests are key to be able to ensure a program is functioning as it should and that there are no errors. |
| **13** | Automation has to do with taking manual processes and making them automated instead. This could be through the use of static code or unit tests. It can be used to help to enforce and comply to security standards. |
| **14** | The DevSecOps pipeline represents a set of processes and tools. It allows developers to cohesively work together to be able to create code productively. It has different steps in the pipeline that help developers to reach the end goal. It is also continuous – which is a key characteristic.  Design phase: static analysis tools as well as built-in IDE tools are used in this phase to help detect errors and potential problems.  Build phase: static analysis tools are used here, and they will give warnings of non-severe issues. If there are severe issues, the code will not compile.  Verify and test: automated analysis tools and automated testing tools are used.  Some tools include: Cppchecker, Parasoft, Coverity |
| **15** | Problems:  Software vulnerabilities that leave sensitive information exposed  Unexpected behavior of code that results in vulnerabilities  Vulnerabilities that are exposed and leave the system unprotected  Solutions:  Defensive coding  Think about security at every step of development  Address security concerns by priority  Use security policies and principles as well as coding standards to the fullest degree  Risks:  Cost and time of having to fix the system if vulnerabilities are exploited  There is the potential of damage to reputation  Loss of trust due to exploitation of vulnerabilities  Benefits:  Secure system or application  High quality system  Maintain user’s trust |
| **16** | As of now this security policy is up to date, and thoroughly addresses security concerns and policies that will help to keep this system as secure as possible  This security policy should be updated if new risks are found, or as new vulnerabilities come up. That was security is maintained to the highest degree possible, and vulnerabilities are kept to a minimum.  Always adapt policies to be used with different languages and platforms to ensure that security policies are followed  Keep re-evaluating every so often, so the policy stays as up to date as possible |
| **17** | Security standards and principles should be followed and implemented to the highest degree possible  Security policy and standards should always be updated to reflect updates and changes  Always consider motivations of hackers, and make sure to stay on top of current hacks going on  Security should be adopted at every step of development  Zero trust is a strategy that can be used along with Triple-A Framework to keep the system secure |
| **18 & 19** | Here are the references I have used. This concludes my presentation for Green Pace. |