# CS 405 Project Two Script Template

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

[**https://www.youtube.com/watch?v=RGg3reZa0ws**](https://www.youtube.com/watch?v=RGg3reZa0ws)

| **Slide Number** | **Narrative** |
| --- | --- |
| **1** | Hi everyone! My name is Robert Lulashi. I am a Marine Corps Veteran and while I am not studying or working on my skills as a developer, I challenge myself by competing in powerlifting. By the end of the summer, I will have earned a bachelor’s degree in computer science with a concentration in software engineering. Today, I will be going over the updated security policy for Green Pace. |
| **2** | The purpose of this presentation to is discuss our updated security policy and how Defense in Depth Is integrated in our policy. By incorporating our new principles and standards, we can reduce the impact and chances of cyber-attacks. |
| **3** | Likelihood and priority are based on different levels that can occur. Level 1 contains priorities 12,18, & 27. This has a high chance of taking place and are of high severity. Level 2 contains priorities 6,8, & 9. These have a lesser chance of happening and have a medium severity. Level 3 contains priorities 1,2,3, & 4. These are unlikely to occur and have low severity. |
| **4** | Green Pace’s 10 security principles are: Validate Input Data, Heed Compiler Warnings, Architect and Design for Security Policies, Keep it Simple, Default Deny, Adhere to the Principle of Least Privilege, Sanitize Data Sent to Other Systems, Practice Defense in Depth, Use Effective Quality Assurance Techniques & Adopt a Secure Coding Standard. |
| **5** | Green Pace’s 10 security standards are: Do not write syntactically ambiguous declarations. Certain syntax can be interpreted by the compiler as either a string or an integer. The ambiguity can lead to unwanted behavior if left unchecked. Do not read uninitialized memory. Trying to read from indeterminate values can lead to undefined behavior such as arbitrary code execution (ACE). If you have ever played a Pokémon game for the original game boy color, ACE can have some unintended behavior. Guarantee that storage for strings has sufficient space for character data and the null terminator. Buffer overflows are common when there is not proper allocation of memory.  Prevent SQL injection. developers should validate all user input, use prepared statements with parameterized queries, employ stored procedures, and follow secure coding practices to ensure the database can distinguish between user input and executable code.  Detect and handle memory allocation errors. Proper handling of exceptions and when to throw them will allow graceful handling to remove unwanted behavior. Use a static assertion to test the value of a constant  expression. This is helpful to catch errors early on in development which helps keep the process running efficiently. Do not leak resources when handling exceptions. Resource leaks can lead to instability in the project and, in severe cases, denial-of-service attacks if an attacker can trigger repeated leaks. Properly deallocate dynamically allocated resources. Failing to deallocate memory leads to memory leaks, which can degrade program performance and eventually cause crashes. Do not confuse narrow and wide character strings and functions. Narrow and wide functions are structured differently and using the incorrect function can lead to buffer overflows and incorrect string lengths. Do not dereference null pointers. By attempting to do so ACE can be exploited, or program crashes can occur. |
| **6** | There are three different types of encryption. Encryption at rest, in flight, and in use. Encryption at rest refers to data being encrypted will not in use. This data is useless unless there is access to the decryption key. Encryption can help during an attack as long as the key remains uncompromised because without the key, the data is still safe within and cannot be accessed. Encryption in flight refers to the transportation of data from one location to another. To safely do this, data must first be encrypted before the transit process can begin. And lastly encryption in use refers to accessing data after using the decryption key. The data is exposed and we must find a way to encrypt it during use to prevent loss of precious data. |
| **7** | * Authentication: This is verifying that the user is the correct user by providing login information and possible additional forms of verification such as Two-Step authentication. Biometric data can also be used to authenticate a user if the resources are allowed to use it. * Authorization: Once a user is verified, only certain permissions will be accessible to them. The principle of least privilege is an excellent use of this. If a user had higher levels of authorization for their account, a hacker could try and exploit this and have much higher levels of access they should not be able to. * Accounting: This is overseeing and monitoring what is occurring with all users. This is another blockade that is in place to ensure security is maintained. If during monitoring it is revealed that someone has given themselves “super” permissions this could flag the system because for what reason was someone given much higher levels of authorization. |
| **8** | Now we will be discussing unit tests. Unit testing is part of the software development lifecycle and is vital in secure coding. These tests, when performed correctly, verify that each section of code is functioning as intended. Also tests are written in a manner that is trying to mimic unwanted behavior and we are screening that the code will not faulter. The first test we will be going over is can resize increase collection. This is first verifying that the collection is empty. If the test confirms this true, then the collection is tested to be of size 0, because its empty. Now that these are both confirmed we resize the collection and then test for the updated size. |
| **9** | The next test is the capacity greater or equal size. This test verifies that when adding entries to the collection that the sizes reflect the changes made. |
| **10** | Can erase clear collection is verifying that after entries are added to the collection, the entries can be removed and are no longer accessible. |
| **11** | Can throw out of range is another test that is conducted and is also vital to secure coding. Exceptions being thrown are helpful when cases are presented in which an error would normally occur. Trying to access values that are outside of the prescribed range will throw an exception. If this is handled incorrectly, then a hacker could try to exploit this area and use ACE. |
| **12** | 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. The security principles can be incorporated within the already existing DevSecOps. |
| **13** | The pre-production phase plays a vital role in laying the foundation for what the future project will look like. The Assess and Plan phase can incorporate the overall architecture and design for the project while revisiting the plan as new threats arise. Principles such as validating input data, heed compiler warnings, and keeping it simple, using effective Quality Assurance techniques, and adopting a secure coding standard are going to be focused on. Once in the production phase aspects like sanitizing data sent to other systems, default deny and practicing defense in depth are now to be focused because there is no indication when an attacker will strike. Being proactive rather than reactive is going to save time and money. |
| **14** | Act Now or Wait? Proactive measures towards security are beneficial because they are more cost effective in the long term. Integration of action is implemented in the DevSecOps already. Waiting is necessary after automation steps are implemented. Constant review of code is part of the automation, but it is unknown when an attack will occur. Overall, this strategy of being proactive saves time and money in the long run but is initially costly in both time and money. |
| **15** | The current policy goes over planning and work during preproduction phase. Actions are identified once the project is launched but specifics on how to handle threats are left open in the respond phase. Detailing how to handle what happens after an attack should be included. Steps on reducing damage or closing off access to compromised data should be addressed. Contingency plans should be implemented in the event of an attack. The saying, “prepare for the worst, hope for the best” comes to mind. Lastly proper training for all developers to raise the standard and also ensure compliance with protocols. |
| **16** | The principles and standards are in place to work with DevSecOps and produce secure code. Training must be routinely conducted to ensure compliance. Overall, keeping things simple is always a best practice because the more regulations are in place, complications and lack of proper communication will lead to data being compromised. This concludes my presentation. |
| **17** | Thank you for your time. |