# CS 405 Project Two Script Template

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

| **Slide Number** | **Narrative** |
| --- | --- |
| **1** | Security Policy Presentation. |
| **2** | OVERVIEW: DEFENSE IN DEPTH:   * Defense in Depth means to make sure that security is priority and has multiple layers to ensure the best protection possible for the company. |
| **3** | THREATS MATRIX:   * The following are the major code standards rankings according to how likely it to happen. |
| **4** | 10 PRINCIPLES   1. ValidateInput Data: Input validation is a crucial security measure that helps prevent various common attacks, like SQL injection, command injection, and cross-site scripting. All unknown data should always be considered malicious and be checked 2. Heed Compiler Warnings: When compiling code the highest level compiler warning should be used to ensure that the compiler detects potential issues, vulnerabilities, or problematic code constructs. 3. Architect and Design for Security Policies: Software should be designed and built with security in mind. Like keeping variables, methods, and functions as protected as possible with data transmissions and using the best security methods available. 4. Keep It Simple: Keep it Simple means to make your developed code as easy to read and maintain by multiple developers to come. By ignoring this and complicating your code can only lead to software issues and security flaws. 5. Default Deny: All systems by default should deny user’s access. This will prevent user ID theft and default admin accounts. 6. Adhere to the Principle of Least Privilege: Principle of least privilege means to grant this minimum necessary permission to user, processes, and systems. 7. Sanitize Data Sent to Other Systems: Data should be checked and sanitized before it is sent to any other system. This is a critical step in maintaining the integrity and security of your systems. 8. Practice Defense in Depth: Defense in depth means to protect the software system in layers, if a layer is to fail there will be more layers to pick of the slack and keep your system protected. 9. Use Effective Quality Assurance Techniques: Using good quality assurance techniques such as automation testing, unit testing, etc. Will lead to reliable software that can meet user expectations. 10. Adopt a Secure Coding Standard: Using a secure coding standard developed for the language that is being used can ensure that your code is better protected from known threats. |
| **5** | CODING STANDARDS:   1. STD-001-CPP: Obey the one definition rule. 2. STD-002-CPP: Do not declare or define a reserved identifier 3. STD-003-CPP: Do not attempt to create a std::string from front a null pointer. 4. STD-004-CPP: Prevent SQL Injection. 5. STD-005-CPP: Properly deallocate dynamically allocated resources. 6. STD-006-CPP: Use a static assertion to test the value of a constant expression. 7. STD-007-CPP: Honor exception specifications. 8. STD-008-CPP: Use valid references, pointers, and iterators to reference elements of a basic\_string. 9. STD-009-CPP: Pass an object of the correct type to va\_start. 10. STD-010-CPP: Do not subtract iterators that do not refer to the same container. |
| **6** | ENCRYPTION POLICIES:   * At Rest: Encryption at rest will protect data while it is stored no matter where that is such as in the cloud, phone, etc. * In Flight: Encryption in flight is the practice of sending data through a secure channel. Should be used whenever possible for any company information being transferred to prevent leaks or unauthorized access. * In Use: Encryption in use is the practice of keeping memory data encrypted even while it is being used. This can help prevent forced crashes that cause memory loss. This should be used on all company data. |
| **7** | TRIPLE-A POLICIES:   * Authentication: Authentication verifies a user's identity credentials. There are several ways to implement this such as username/password logins, digital certificates, etc. All users should be verified to prevent theft or unauthorized users. * Authorization: Authorization is used to define the level of access to a system of files. Users will be checked for authorization before gaining access to any of the company files. * Accounting: Accounting is the records, or log files that detail what users have been doing. This would keep track of what is going on such as when a new user is created, whose accessing what, etc. |
| **8** | Unit Testing:   * The following are unit tests that test different properties of a collection object. This will show a variety of examples on how unit test work and why they are effective. |
| **9** | Verify adding a five values to a collection:   * This test here verifies that five values can be added to an empty collection |
| **10** | Verifying that max size is greater than entries:   * This is a unit test checking if the max value is greater than the existing entries |
| **11** | Verify that resizing increases the collection:   * This checks to make sure that resizing increases the collection. |
| **12** | Verify that clear works:   * The following is a unit test checks to make sure that clear erases the collection |
| **13** | TOOLS:   * The DevSecOps pipeline is a integrating security measures into each step of the DevOps toolchain. This will allow security to become an important and key part in an ever-growing technological world. This will also make sure that the company is as protected as possible from outside threats. * There are a variety of tools that can be used to help with providing security coverage. Google testing framework for unit testing C++ applications. CPPChecker for static code analysis. Theseare just a few of the software tools for development. |
| **14** | RISKS AND BENEFITS:   * There are many benefits associated with adopting a security first approach. A big benefit is consumer confidence. The more trusting the consumer is to your product the more likely they are to use your company. * There are many risks involved in adopting a security policy. Just having user authentication isn’t enough. You need be cautious of cyber attacks because in today's world they happen all the time. |
| **15** | RECOMMENDATIONS:   1. Check all compiler errors in code and run the code through a static code analysis program. This will verify the risks in the code and help you with fixing them. 2. Enact account authorization and authentication. This can make sure that when one user is hacked it does not compromise the whole system. 3. Security training for all employees. Keeping up with training for staff will ensure they know the proper precautions when dealing with possible attacks. |
| **16** | CONCLUSIONS:   * Defense in Depth: This is necessary, having multiple layers of coverage will help prevent attacks and mitigate slight attacks. * DevSecOps: Moving to a DevSecOps model will help ensure that security is a priority in all rollouts and development. * Authentication and Authorization: Making sure you validate who uses the systems and what can access are very important for protecting data. |