# CS 405 Project Two Script Template

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

| **Slide Number** | **Narrative** |
| --- | --- |
| **1** | Hello everyone. My name is Ashley Lien and I will be giving the security policy presentation. |
| **2** | Defense-in-Depth is a strategy which uses multiple layers of security measures to protect an organization’s assets. When using multiple security measures, it decreases any chances of attacks against the network or even hardware on the physical layer. Since it utilizes multiple layers of defense, if one defensive measure is compromised, there are other layers to protect it. The security policy has 10 coding standards to help protect Green Pace. This includes C++ coding standards; authorization, authentication, and auditing standards; and data encryption standards. |
| **3** | Looking at the threats matrix, the standards that are likely are STD-002-CPP, STD-003-CPP, STD-004-CPP, STD-007-CPP, STD-008-CPP, and STD-009-CPP. These standards consist of data value, string correctness, SQL injection, exceptions, and memory protection. The standards that are priority are STD-001-CPP and STD-005-CPP. These standards consist of data type and memory protection. There are no low priority coding standards in this current security policy. Lastly, the standards that are unlikely are STD-006-CPP and STD-010-CPP. They consist of assertions and data value standards. There are multiple automation tools that can be used to detect these standards such as, CodeSonar, Parasoft C/C++ test, and LDRA tool suite. They use checkers to determine if the standards listed are in the written code. |
| **4** | The 10 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, and adopt a secure coding standard. Next, we will be looking at which coding standard applies to which principle. The coding standards that apply to the validate input data principle are STD-001-CPP and STD-004-CPP. The coding standards that apply to the heed compiler warnings principle are STD-006-CPP and STD-007-CPP. For architect and design for security policies, the coding standards are STD-005-CPP and STD-008-CPP. For the keep it simple principle the coding standards are STD-003-CPP and STD-009-CPP. Lastly, use effective quality assurance techniques coding standards are STD-002-CPP and STD-010-CPP. Not every principle was used for this particular security policy, however they are still very important when it comes to security. |
| **5** | The coding standards are listed in priority order. I chose this order when considering likelihood, severity, and the remediation cost. STD-008-CPP is do not access freed memory. STD-009-CPP is do not attempt to create a std::string from a null pointer. STD-004-CPP is exclude user input from format strings. STD-002-CPP is ensure that division and remainder operations do not result in a divide by zero errors. STD-003-CPP is do not attempt to modify string literals. STD-007-CPP is handle all exceptions thrown before main() begins executing. STD-005-CPP is allocate sufficient memory for an object. STD-001-CPP is ensure that integer conversions do not result in lost or misinterpreted data. STD-006-CPP is detect errors when converting a string to a number. Lastly, STD-010-CPP is do not cast to an out of range enumeration value. |
| **6** | For the encryption strategies, I will summarize the policies for encryption in flight, at rest, and in use. Encryption in rest provides protection for stored data. It prevents an attacker from accessing unencrypted data by making sure it is encrypted when it is stored. It should be used as an extra layer of protection. If someone were to obtain physical access to the data, they would not be able to access the data itself as easily. Encryption at flight is the encryption of data that moves over a network. This protects the data while it is being transmitted. It should be used so the data is protected in case it gets intercepted. Encryption in use encrypts data that is in use when accessed by a user or application. This helps protect against cyber attacks since data that is in use is the most vulnerable form of data. |
| **7** | The triple-A framework policies are authentication, authorization, and accounting. Authentication is when a user can prove who they are providing information. This information can be a username and password, keycard, or biometrics. This is to make sure only authorized users are allowed certain levels of access to protect against attacks. Authorization is what determines what the user has access to. For example, there would be user privileges or administrator privileges. The administrator would have a higher level of authorization. This helps protect against attacks but can also prevent negligence. Accounting keeps track of user activity while they are logged in. Accounting may be used to analyze user trends and audit user activity. This policy applies because if a change was made to the network or program there would be a way to see which user was making what change at what time. |
| **8** | In the example shown, unit testing can be conducted to check if specific units of code comply. If the code is in compliance, it will show as passed. If the code does not, it will show as a failed test. |
| **9** | This is a DevSecOp diagram. Automation will be used for the enforcement of and compliance to the standards defined in this policy. Green Pace already has a well-established DevOps process and infrastructure. |
| **10** | Automation can be done in pre-production and production phases. The process to automate enforcement of the standards can be done in the design, build, verify and test phases of the pre-production. This will ensure that the standards are being enforced throughout the development of the project. Automation tools can be used during development. In the production phase, they can be used to maintain and stabilize. This will help detect vulnerabilities. |
| **11** | The problems were seen during the risk assessment, along with the severity, likelihood, and remediation cost. I believe that the benefits to correcting these problems outweigh the risks if we were to wait. This is because a lot of the problems were likely with a high severity. The only problem is the remediation cost. The steps to be taken should be to check with finances and see how much it would cost to fix the risks based on priority order. |
| **12** | There are steps that can be taken to prevent security threats. First, I would try to think about the types of potential vulnerabilities that could occur during the planning phase. For example, I would make sure to have SQL injection and buffer overflow prevention. This protects against types of attacks where the user can enter an input. This gets done during the development stage. Unit testing and code analysis can be conducted during development and after. The developer should go through and check the list if there are any warnings to correct them to prevent attacks.  As previously mentioned, unit testing will help ensure that security is addressed before an issue is discovered. By conducting unit testing, the developer will be able to discover potential threats before a hacker can have the chance to make the security threat known. This can be done by compiling a Google unit test fixture with positive and negative tests. This will ensure that the code is able to meet the requirements, functionality, and best practices. |
| **13** | All the standards should be adopted moving forward. They all aid in the protection and add to defense in depth for Green Pace. The coding standards protect against attacks and issues that could occur during future projects. They should all be followed in the future. |
| **14** | Here are the references used. Thank you for watching my presentation. |