# CS 405 Project Two Script

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

| **Slide Number** | **Narrative** |
| --- | --- |
| **1** | Hi, my name is Jonathan Ramirez, and I will be presenting my security policy in which I will be explaining a proposed policy guide for Green Pace. |
| **2** | In the age of massive data manipulation, there are constant threats that aim to exploit this data to either harm the users, the company, or to just show how vulnerable a company’s software security is. This is why Green Pace needs a security policy that highlights the most common and the most dangerous vulnerabilities as well as how to mitigate them using standard C++ coding practices. |
| **3** | From highest severity, we have Memory protection. It is the highest because low memory protection is an easy target for cyberattacks and because it will result in a cascade of other problems if memory is compromised by hackers.  We then have:  STD-006-CPP  STD-010-CPP  STD-001-CPP  STD-007-CPP  STD-004-CPP  STD-008-CPP  STD-009-CPP  STD-003-CPP  Finally at the bottom of the list we have Data values. Data values are not much of concern for data vulnerabilities. They are more for when trying to debug code as implementing standard data values like proper naming conventions can facilitate in reading code especially when debugging occurs. |
| **4** | Validate Input Data   * STD-006   Heed Compiler Warnings   * STD-006   Architect and Design Security Policies   * STD-006 * STD-005   Keep It Simple   * STD-008 * STD-009   Default Deny   * STD-007   Adhere to the Principle of Least Privilege   * STD-005   Sanitize Data Sent to Other Systems   * STD-010   Practice Defense in Depth   * STD-004 * STD-005   Use Effective Quality Assurance Techniques   * STD-002 * STD-003 * STD-010   Adopt a Secure Coding Standard   * All Coding standards proposed |
| **5** | Data Type  Data Value  String Correctness  SQL Injection  Memory Protection  Assertions  Exceptions  Code Complexity  Pointer Naming  Close Files |
| **6** | Data that is not being used should be stored in a secure database. This data should be encrypted. For example, encrypted mail folders, encrypted file system, or self-encrypting hard drives  Data that is being transferred should be protected using HTTPS, SSL, TLS, or FTPS.  Data that is being used should also be protected through encryption methods. One example is In-Use encryption which protects data from the beginning all the way to where it is being used. It simply keeps data encrypted while it is being used for things like computations. |
| **7** | Authentication ensures users are who they really are like account creation, email verification, and two-step authentications.  Authorization is the idea of users being allowed only the data they are allowed to. For example, an employee should not have the same access rights as an executive.  Accounting is monitoring what everything is happening around the company’s networks, web applications, systems, etc. Users activities are should be constantly monitored. |
| **8** | 1. This vulnerability checks if two sets are not equal, if they are not then the test passes resulting in a negative test. 2. This unit test checks if a list of numbers contains more than 5 numbers that are divisible by two, if it finds 5 or more the tests pass resulting in a positive test. 3. This test ensures the clear() method actually erases a list of numbers by comparing the starting list to zero or null. 4. The unit tests check to see if the maximum size of the list is equal for 0, 1, 5, and 10 entries. |
| **12** | Sometimes code becomes repetitive and if this code is being developed or maintained by humans, there could be errors introduced. In DevOps, automation is the process of automating repetitive code or processes. It drastically reduces human errors because no human interaction is involved. |
| **13** | 1. Jenkins is an open-source automation server that is customizable. It promotes CI/CD by automating the build, test, and deploy stages. 2. Ansible is used in the deployment stage. It is used for tasks such as IT automation, Network automation, and security automation. 3. GitLab CI/CD helps automate the building, testing, and deployment stages of the DevSecOps. 4. Finally, Bamboo helps automate the build stage as well as the test, and releases stages. |
| **14** | Problems   * The problem with leaving security to the end is that it will be time consuming and costly because a lot of vulnerabilities will have to be fixed at the end of the cycle as opposed to throughout the stages of the cycle.   Solutions   * A solution is to start implementing security best practices from the beginning. There should also be educational seminars throughout the cycle to teach the developers about security.   Risk or Benefits   * Leaving security to the end could result in time being wasted as well as money from the company. It can also result it dangerous loss of user personal data that might result in legal actions against the company. * Implementing security practices from the very beginning ensures minimal bugs and errors to correct when unit testing comes around. It will also speed up the development process and will ensure a system is well designed. |
| **15** | No security policy is 100% effective, there are always gaps that need to be patched. There are always new threats emerging, thus a policy will never be complete, it will only grow. So, emerging threats should be analyzes and use this information to modify the policy. Other not so notable threats are Privileged access and third-party access |
| **16** | I feel like all principles are important but the most crucial to implemented to help prevent future problems are practicing defense in depth, adopting a well secure set of coding standards, keeping code simple and clean, and validating input data. |
| **17** |  |