# CS 405 Project Two Script

**https://youtu.be/bOa\_Q6fNs60**

| **Slide Number** | **Narrative** |
| --- | --- |
| **1** | Hello, my name is Jesse Troutt and this is my presentation on the Green Pace Security Policy. |
| **2** | As a general overview, this policy uses security principles to identify coding standards to follow in order to provide defense in depth for coding procedures. |
| **3** | Here we have the threats matrix. It shows a matrix of likely, unlikely, priority, and low priority. Higher priority threats should take precedence, but lower priority threats should not be ignored. As we can see, some of the lower priority threats are actually more likely to occur, so they should also be considered. |
| **4** | These are our 10 principles. Validate input data, which is used by the 5th and 8th standard. Heed compiler warnings, which is used by the 1st and 6th standard. Architect and Design for Security Policies, which is used by the 3rd and 4th standard. Keep it Simple, which is used by the 7th and 10th standard. Default deny, which is used by the 8th and 9th standard. Adhere to the Principle of Least Privilege which is used by the 10th standard. Sanitize Data Sent to Other Systems, which is used by the 2nd and 3rd standard. Practice Defense in Depth, which is used by the 2nd and 5th standard. And Use Effective Quality Assurance Techniques, which is used by the 1st and 6th standard. |
| **5** | There are a total of 10 coding standards used in this security policy. Don’t qualify a reference type with const or volatile. Do not cast out-of-range enumeration values. Strings must have sufficient storage space for characters. Check the range of element access. Do not try to read uninitialized memory. Understand when to use assert. All exceptions should be handled. Use valid pointers. Clean up resources when handling exceptions. And do not rely on side effects of unevaluated operands. |
| **6** | The encryption policies involved include encryption in rest, encryption at flight, and encryption in use. Encryption in rest refers to data that is in storage or not being used. At flight refers to data currently being transmitted such as emails. In use is meant to protect data being currently used or edited. |
| **7** | Following the encryption policies are the Triple-A policies, or the Authentication, Authorization, and Accounting. Authentication can be seen as a login, it authenticates that the user is who they say they are. Authorization is the user’s ability to interact with data, and Accounting is the proper recording of data. This usually refers to keeping log files of user activity like file changes and downloads. |
| **8** | Here we have Unit Testing. The next four slides will be the different tests run and what they achieved. This slide contains the screenshot showing that all unit tests have been run successfully. |
| **9** | This unit test is meant to show that by using the reserve function, it will increase the capacity but not increase the size. |
| **10** | This unit test is used to verify that an out\_of\_range exception is thrown when calling at() with an index out of bounds. This is a negative test which means that for the test to run successfully, the code would have to throw an exception. |
| **11** | This is another negative test. It’s used to verify that calling at() will throw an out\_of\_range exception when the number used is not a whole number. |
| **12** | This final test shows that using multiple add\_entries() will work as intended and increase the collection size to match the added entries. |
| **13** | The automation summary is meant to enforce the standards in this policy into the DevOps pipeline using automation. This ensures minimal user involvement while staying in compliance with the outlined policy. |
| **14** | There are a multitude of tools used to help enforce standards within this policy. DevSecOps is meant to integrate security into the software development lifecycle. CodeSonar is used to assist with finding and fixing bugs. Parasoft C/C++test is an integrated set of tools for testing, analyzing, finding defects, and measuring quality of source code. Polyspace Bug Finder identifies run-time errors and other defects in code. |
| **15** | The policy defined has many benefits and a few risks involved. To be clear, the strategies within start with motivation. Whatever is believed to motivate an attack is the first to be defended. This policy focuses on that first. This means protection from known sources and known issues. The risks are that this policy is not airtight. It should be assumed that there are vulnerabilities that need to be addressed and this policy should be thought of as a starting point, not an absolute solution. |
| **16** | Software security is a constantly evolving problem. There will always be future issues to consider and new standards to implement. Use this policy as a starting point. Continue improving and adapting. Identify new standards to adopt and never assume software is completely secure or a policy is perfect. |
| **17** | To conclude, the standards defined within this policy identify many vulnerabilities to protect against. But one area not defined is File usage. File input and output should have defined standards added to the policy in the future. This includes closing files when they’re no longer needed, excluding user input from format string, and not copying a file object to name a few. |
| **18** | These are the references used throughout the development of this policy and presentation. |