# CS 405 Project Two Script

**Bradley M. Emerson CS-405 Project Two: Security Policy Presentation 4/19/2025**

[**https://youtu.be/3VvHOX3UUGE**](https://youtu.be/3VvHOX3UUGE)

| **Slide Number** | **Narrative** |
| --- | --- |
| **1** | Hello everyone. Today I’ll be walking you through our Green Pace security policy presentation. This guide lays the foundation for secure software development using 10 essential principles, a threat matrix, encryption policies, and the Triple-A security model. This slideshow presentation may be painful to get through, but know that there is free pizza waiting upon completion. That being said, I would appreciate your undivided attention throughout the presentation, if you help me out by learning this material, we can all get back to business. |
| **2** | We will be using a strategy called “Defense-in-Depth”, which I’m sure most of you are familiar with. The security policy we’ve developed sets forth some essential principles illustrating 10 examples using rules from STD-001-CPP. |
| **3** | This threat matrix helps illustrate how we classify certain vulnerabilities into separate threat categories. You will notice the category of plain “Priority” absent here, as none of our rules specifically fit the bill. These rules were classified into these threat categories using a combination of criteria such as remediation cost, likelihood of exploitation, and numerous other factors. The bottom line is that these issues all need to be addressed, but in order of importance. |
| **4** | From input validation to quality assurance techniques, these 10 principles guide every line of code we write. They are the lynchpin of our policy, and should be used throughout the entire Software Development Life Cycle. |
| **5** | We’ve prioritized 10 CERT rules. While top-tier threats relating to rules like FIO30-C and STR31-C are more likely and more severe relative to others, none of these rules are optional—they all help enforce reliability, maintainability, and security. |
| **6** | We encrypt data at rest, in transit, and in use. That means AES-256 for storage, RSA for transmission, and AMD SEV for volatile data in remote virtual machines—ensuring confidentiality and integrity end-to-end. |
| **7** | Authentication verifies who you are. Authorization limits what you can access. Accounting logs everything for audit and review. This is our three-layered approach to control and monitor system use. |
| **8** | We demonstrated several of the aforementioned principles and rules via unit tests, as can be seen in the following slides. You will notice a recurring theme of not accessing freed memory- in the military we used to call this a “foot stomp” moment, perhaps some of you get the reference. In this slide, we run a test to ensure that capacity of a vector is increased using the reserve function. |
| **9** | Here, we test whether or not we can add an entry to an empty vector. Spoiler, we could. |
| **10** | This test checks to make sure that order is preserved in a vector when using the push\_back function. |
| **11** | (Accidentally left narrative blank and realized during presentation, performed off-the-cuff and inserted this comment post-production) |
| **12** | This is one of two “negative” tests, meaning that the test passes when we achieve forcing an error, and in this one we throw an exception by attempting to access an element of a vector that doesn’t exist. |
| **13** | And here, we attempt to convert the letters “NAN” to a number, which doesn’t go very well, as expected, and an exception is thrown. |
| **14** | We will be using automation throughout our development processes to ensure that security is integrated in every phase possible (hint, it’s all of them.). |
| **15** | We use Cppcheck for static analysis, Visual Studio’s compiler for syntax-level issues, and Google Test for unit validation. These tools automate detection and reduce manual error. There will be many tools available for your use regarding several different rules, but an exhaustive list would be absolutely massive, as you can imagine, so please refer to the security policy and the specific tools listed there, or even better- visit SEI CERT STD-001-CPP via the link provided in this slide show and see each rule for a list of tools used in detection and prevention. |
| **16** | Waiting to secure code leads to rework, wasted time, and higher costs. Proactive secure coding reduces bugs, improves product quality, and aligns with modern DevSecOps practices. |
| **17** | This policy is a foundation - but security doesn’t stop here. Daily patching, routine audits, and vigilance must become part of our team’s ethos. Remember Equifax? A single missed update to a singular framework caused catastrophic damage. |
| **18** | In closing, these standards aren’t just rules, they’re a mindset that we all need to adopt. Let’s apply them consistently, learn continuously, and hold ourselves to the highest standard. Security isn’t a task, it’s our responsibility. Our livelihoods, and the livelihoods of our customers depend on it. |
| **19** | (No narration, references page) |