# CS 405 Project Two Script

Sawyer Prestwood, 04/16/2021, Project Two: Security Policy Presentation, <insert link>

|  |  |
| --- | --- |
| **Slide Number** | **Narrative** |
| **1** | Hello. Today I will be highlighting key points of the new Security Policy written for Green Pace. |
| **2** | The basis for this policy showcase is to explain, at a high level, the security risks currently faced by the organization, policies and standards that can mitigate commonly faced development issues, and security practices to ensure the digital safety of both the company and our customers. |
| **3** | Here is a simple threat matrix concerning the 10 security policies identified in the outline. As you can see, the largest security threats come from string and memory management, specifically in the use of C++ in applications. While these are all important, including the “unlikely” ones, we can use automated tools to detect and prevent these issues. |
| **4** | There are at least 10 principles of secure coding to be focused on. Here, we can see them in order, as shown in the outline. Under several of them, you will see the coding standards which best apply to the principle. While not all principles are matched with a single standard, these represent the best practices to use in all coding projects for the company. |
| **5** | Here are the coding standards, labelled and organized by threat level. You can again see that string and memory management top the list, as they tend to cause the most and longest-lasting damage to systems. The security policy paper outlines in more detail what each of these standards are, coding practices, and code to avoid. |
| **6** | There are three states at which data needs to be protected, which essentially covers *all* data at all times. At rest, where data is stored for future use; In Flight, or “in transit”, where data is being sent to or from the system, either externally or internally; finally In Use, where data is being used by the system either to show a user what they need, or to verify account credentials. |
| **7** | Triple-A Framework stands for Authentication, Authorization, and Accounting. Simply put, Authentication is verifying you, or the user, are who you say you are. This can be done using verified devices, or two-factor backup codes. Authorization focuses on permissions; can the person in the system access what they are trying to access? Typically, we follow a rule of least privilege to prevent unauthorized, high-level access to the system. Finally, Accounting, or Auditing, is the act of logging *everything* on the system. No system is fully secure, so having as much information about an attack as possible will help prevent repeat attacks. |
| **8** | Here are a few screenshots of a unit test testing the memory-management functionality of a container in C++. As you can see, the test passed using Google’s Unit Testing framework, more specifically with the ability to allocate memory without actually creating the object. |
| **9** | This is the DevSecOps pipeline, a more streamlined and security-focused version of the DevOps pipeline already in use. There are several places where we can add automation to help lighten the load on developers, as well as keep us up-to-date of the goings on in the system. Specifically, in the “Verify and Test” stage, we will automate the use of several external tools to check for security vulnerabilities, dependency checks, and policy compliance. |
| **10** | Here is a brief overview of what exactly a DevSecOps pipeline is, and further explanation of some of the tools that are recommended for the specific security polices written for Green Pace. Clang and Astree are just two of many that can detect vulnerabilities in a code base. |
| **11** | There are always risks when implementing new technologies, more so when developing a custom-built security system. Firstly, the addition of automation and encryption should protect the data, and alert us of anything that goes against our policies. While the threat is not serious, it is important to implement the best security practices at all times. Should you choose to wait to implement these policies, you may experience issues after the product is released that are harder to manage. Should you redesign your development pipeline as suggested, however, you can get a head-start on detecting, preventing, and addressing security vulnerabilities in the product. |
| **12** | Obviously, I would recommend encryption be *everywhere*. This is one of the most common mistakes companies make, and should there be a security breach, unprotected data is just too easy to steal. The Triple-A framework is also important, both for the system and as an explanation to users. For example, the eBay hack in 2014 was a direct result of improper authentication. Or rather, the exploitation of authentication to access system-level resources and data. Unit testing is an important practice to ensure each piece of code functions as expected before integrating into the system. |
| **13** | While all standards should be adopted to further improve the security of both the systems and applications, special attention should be paid to use of strings, buffers, and memory management especially when writing in C++.  Log files, or the “Accounting” aspect of the Triple-A Framework is also important to consider. Attacks *will* happen, but how we respond depends entirely on the information we can gather about the vulnerability. |
| **14** | Here are just a few references to the articles and quotes used in this presentation. Thank you for your time, and I hope you consider implementing these security polices company-wide. |

**Bibliography**

Nangare, S. (2019, October 10). *DevSecOps pipelines and tools: What you need to know*. Opensource.com. https://opensource.com/article/19/10/devsecops-pipeline-and-tools

Ragan, S. (2014, May 21). *Raising awareness quickly: The eBay data breach*. CSO Online. https://www.csoonline.com/article/2157782/security-awareness-raising-awareness-quickly-the-ebay-database-compromise.html