# CS 405 Project Two Script Template

Companion document for CS-405 Project Two Presentation EFARKAS located at:

[**https://www.youtube.com/watch?v=sR5gHaX060E**](https://www.youtube.com/watch?v=sR5gHaX060E)

| **Slide Number** | **Narrative** |
| --- | --- |
| **1** | Hello, everyone. I’m Eric Farkas, and I will highlight the key points of the new security policy designed for Green Pace. |
| **2** | Green Pace continually faces security risks. This policy explains, at a high level, the security risks plus the policies and standards that can mitigate development issues. It also outlines security practices that help ensure the company's and its customers' digital safety. |
| **3** | Let’s begin with the threat matrix, including the 10 security policies identified in the outline. As you can see, the largest security threats come from input string and memory management techniques used within C++-developed applications. Many identified threats exist due to the lack of code testing with automated detection tools. |
| **4** | This security policy focuses on ten principles of secure coding. They are all important, but they are listed in the order of which they should be implemented into your processes. Most principles are supported by standards. Not all standards apply to the work done at Green Pace. Let’s look at the ones that do. |
| **5** | Here are the coding standards, organized by threat level. You may notice that input validation, strings, and memory management are significant threats. This is because their damage can be long-lasting and time-consuming to fix. The security policy document provides additional details regarding each standard and code, as well as examples of best and bad practices. |
| **6** | This information covers the three states of data and how each needs to be protected. The goal is to always protect data while it transitions from state to state. At rest data is data stored for future use. In-flight data is data on the move or in transit. This data is being sent to or received from another system. In-use data is data being currently transacted against. Typical actions are read, write, modify, and delete, but in-use data is not limited to those actions. |
| **7** | The Triple-A framework is an acronym for Authentication, Authorization, and Accounting. Authentication is the first step and is about verifying the user is who their credentials say they are. This can be done using verified devices, or two-factor methods. Authorization focuses on user permissions. The available resources and actions a user can take are granted by authorization. Typically, the rule of least privilege is implemented to prevent unauthorized, high-level access to the system. Accounting is the process of logging the actions of a user on the system. Implementing audit trails helps to identify problematic user actions and identify a user account that has been compromised. |
| **8** | Unit testing is a mostly automated method of ensuring that small sections of code comply with security policies and best practices. Assertion testing determines whether the code provides positive or negative results when provided with certain criteria. |
| **9** | Large combinations of assertion tests can be combined into a single unit test that all code will be subjected to. This aids testers, or QA in ensuring that each piece of code is compliant with the security policy and criteria within using automation. This enables more time to be spent on elements requiring manual tests. |
| **10** | This is the DevSecOps pipeline. It simply adds a security focus to the existing DevOps pipeline. There are many opportunities for automation. Some of the previous unit test examples fit nicely within the verify and test stage. External tools are also available to check for security vulnerabilities, dependencies, and policy compliance. |
| **11** | In case the graphic was intense or hard to interpret, this is a more classic definition of the DevSecOps pipeline. Below the definition are several external tools that may prove helpful in achieving the new security policy. Tools are often best selected by the teams that use them, which is why I provide examples. |
| **12** | Implementing a security-focused development strategy and business systems is a fine balance between risks and rewards. Security is important to Green Pace and its customers. This comparison shows the efficiency and security gains of implementing the new policy compared to the risks of not doing so. |
| **13** | Communication and training are just as important as implementing a new security policy. Everyone in the organization must be aware of new technology and equipped to identify threats. Social engineering can defeat some of the most sophisticated security architectures. Developing and testing for this environment will also involve training and practice implementation. Attack vectors and methods are constantly evolving, and the developers keeping systems safe must also constantly evolve. |
| **14** | While all standards should be adopted to further improve the security of systems and applications, special attention should be paid to string, buffer, and memory management, especially when writing in C++. Log files or the accounting aspect of the Triple-A Framework should be considered. Successful attacks may happen, but recovery and prevention are heavily predicated on determining the root vulnerability. Having “bread crumbs” to follow greatly simplifies this analysis. |