# Gabrielle Maitland

August 18, 2024

Project Two: Security Policy Presentation

# CS 405 Project Two Script Template

Video link: <https://youtu.be/GzXIwHDK_2k>

| **Slide Number** | **Narrative** |
| --- | --- |
| **1** | Welcome! My name is Gabrielle Maitland, and I will be taking us through a guided presentation on my proposed Security Policy for Green Pace. |
| **2** | At Green Pace, our security mission is defense in depth, a secure coding technique that involves using multiple methods of defensive cybersecurity practices in order to protect a single system. That is why we will be exploring a streamlined security policy that provides insight and documentation on existing coding and architectural vulnerabilities. We will also explore options in automation, and moving to a DevSecOps pipeline to improve security. |
| **3** | Let’s explore our threat matrix, a visual representation of instances categorized by likelihood and priority. Likely instances are incidents that are expected to happen more frequently, such as accessing freed memory, not sanitizing data before passing it to other systems, and copying data to a buffer not large enough to hold that data. These three instances are also of high priority due to the potential damage to infrastructure they can cause. These priority items should be addressed first and foremost.  There are also unlikely incidents, such as assigning multiple definitions to one function, failing to understand the termination behavior of abort() and assert(), and leaving files open when they are no longer needed. Again, these three instances are of low priority, as they are less likely to wreak havoc on a system and do not require immediate attention. |
| **4** | There are ten core security principles that should underscore a developer’s work when creating code, as they will contribute to having a highly secured system. These principles are to validate input data, heed compiler warnings, architect for security policies, keep it simple, default deny, adhere to principle of least privilege, sanitize data sent to other systems, practice defense in depth, use effective QA techniques, and adopt a secure coding standard. In the table before us, we can see that these principles are also linked to specific coding standards. |
| **5** | Here we have a ranking of the ten coding standards outlined in the security policy. Some of these items felt equally important to me, but for the sake of simplicity, they are ranked from 1-10. Let’s take a closer look at the importance of the top items. Scrubbing data that will be passed from system to system defends us against things like injection attacks. Sufficient storage for strings prevents buffer overflows. Reading uninitialized memory can result in unexpected behavior. Referencing freed memory can result in exploitable vulnerabilities. Each of these standards were chosen due to their importance, severity, and likelihood. |
| **6** | Encryption policies are important to implement, as we are trusted with data from internal and external clients. With that being the case, we should encrypt data in every stage of its life. Encryption at rest describes data that is not in use, or in storage, being encrypted in order to reduce bad actors from having access to sensitive information if attacked. This policy applies to Green Pace, as we should ensure that if our defenses fall and our customer information is breached, the attackers would not be able to decrypt the information due to not having the means or a key.  Encryption in flight refers to data that is moving from one system to another. Data in motion should also be encrypted so that if Green Pace is attacked and our actions are intercepted by a bad actor, they will not be able to read or use the data stolen.  Encryption in use is defined as encrypting data even if actively in use. This policy applies to Green Pace as we hold the trust of our clients when using or changing the data they have provided to us. This ensures that at all times, regardless of the circumstance, data remains hidden and encrypted to anyone without the key. |
| **7** | Much like Encryption Policies, Triple A Policies are also important to ensure proper usage measures for every actor in our system. Authentication refers to validating that a user is who they say they are, whether through two factor authentication, log in information, etc. This policy applies to Green Pace as we should authenticate all users, both internal and external, before allowing any access to any system. This prevents bad actors from gaining entrance to our systems.  Authorization describes checking credentials to see what actions a user is permitted to perform. This relates to the principle of least privilege, a security process outlined earlier in this document. This principle applies to our organization, as we want to ensure that excessive authorization levels are not granted to just any user, and rather granted to those who have the most urgent and necessary needs. For example, if someone who is intended to handle customer relations has the ability to change a database storing sensitive financial information, we may need to reconsider authorization levels.  Accounting is defined as keeping a log of actions that happen in the system. This typically includes who, what, when, where, and how. This audit trail ensures that if a file is accessed, there is a way to review the details of the change, and thus take effective action. This policy applies to Green Pace, as it increases mitigation efforts if anything negative were to happen. |
| **8** | Unit testing is important to make sure that each part of your code works as intended, which aligns with principle number 9: use effective Quality Assurance techniques. Let’s take a look through a few unit tests using Google Test suite. All of these tests passed. This first test was intended to create a test to verify adding a single value to an empty collection. |
| **9** | Creating a test to verify reserve increases the capacity but not the size of the collection. |
| **10** | A test to see if pop back works by removing one entry. |
| **11** | A test to see if an error throws if you try to reserve over the max size. This is a negative test. |
| **12** | Automation will be used for the enforcement of and compliance to the standards defined in this policy. While Green Pace already has a well-established DevOps process and infrastructure, we can improve on these processes by introducing automated enforcement of security standards to evolve our infrastructure into DevSecOps. Automation can be implemented beginning with the build stage of preproduction. To achieve a secure build, we can use the aforementioned automation tools in this policy to begin checking for security vulnerabilities. During the verify and testing stage, we can introduce continuous integration tools along with our security automation to help automate the process of development, to catch any quiet issues, and to ensure that our new code introductions are stable. |
| **13** | The DevSecOps pipeline adds additional focus to Security in the Software Development Lifecycle through continuous integration and delivery. With security being an added focus, there are beneficial tools that can be used to make the workflow easier than before.  In Green Pace’s practice of using CPP, dev tools like CPP Check can assist in facilitating the proper and secure development of CPP code.  Good examples of reliable CI/CD tools would be Jenkins or Travis CI.  Automated testing tools such as Selenium, Cypress, or Katalon can also be introduced to automate testing depending on our needs. |
| **14** | It is important to implement secure practices immediately. The longer we wait, the more likely we are to face attack from bad actors. This can lead to loss of trust, capital, clientele, and more. Waiting to secure our code also gives us the opportunity to make our code more and more convoluted, thus making it difficult to layer in security in the end. However, we can benefit from securing our code by having trust that every developmental effort going forward will follow a secure standard. This will help us to continue to elevate the levels of trust not only amongst ourselves, but also with our clientele. It will also aid in future endeavors, as we will have code that we can continue to secure when the latest and greatest finds in cybersecurity become public. |
| **15** | I would recommend that Green Pace invests in robust DevSecOps training for its developers, with increased attention given to the security team. This will align all of our employees on the best practices, and will equip our security team with more resources to protect our company.  We should invest and implement some of the aforementioned security tools in order to increase work efficiency and heighten programming security. Regular audits on our current practices should remain, but we should implement a monitoring service – whether in house or third party – to identify potential attacks and thus mitigate and solve them. |
| **16** | In conclusion, Green Pace is doing relatively well with the processes that we use, but we have opportunity for growth and security. In order to prevent future problems, we should adopt the Ten Core Security Principles, Ten CPP Coding Standards, a DevSecOps pipeline with automation, Triple A Methodology, and Data Encryption Policies. These practices will take Green Pace’s security to the next level, creating a safer environment in all of our work and for everyone involved. |
| **17** | Here is a list of referenced materials. Thank you for listening! |