# CS 405 Project Two Script

Alexis Indick

June 18th, 2023

Project Two: Security Policy Presentation

Link: <https://youtu.be/6B4Roknu6gY>

| **Slide Number** | **Narrative** |
| --- | --- |
| **1** | Hello, I’m Alexis Indick and today I will be presenting Green Pace’s security policies and standards. |
| **2** | In this first slide, we can see this very important model to keep in mind when performing security measures. Defense in depth is meant to have many layers to protect the application or perhaps a system someone is using. By performing proper defense in depth techniques, we can prevent harmful attacks. |
| **3** | On the next slide, we have a security threat matrix. This matrix is useful to keep in mind when it comes to remembering what possible threats are out there. The “Likely” threats are the ones that will most likely happen and should be watched out for. The “Priority” threat means that a vulnerability is very important to take care of immediately. “Unlikely” means that a threat will most likely not happen. “Low Priority” means that the vulnerability is not too important to fix immediately but should be fixed asap. |
| **4** | This side shows the 10 Principle of Security every developer should keep in mind. Number 1 is to validate the data and when we validate data, we make sure that hackers cannot input malicious phrases or files that could expose vulnerabilities in a system. Number 2 is heeding compiler warnings, and this is quite important since ignoring compiler warnings could be fatal in the way that bugs could be within the code and exploited later by hackers. Warnings help us prevent disaster. Number 3 is to be kept in mind since we need make sure we are designing soundproof security policies for products. Number 4 is important since if we make an application too large and confusing, it can be harder to go back and figure out where issues are if we don’t keep things simple. Number 5 involves not giving allow permissions automatically to users. Number 6 is about giving the least amount of permission to users so that they don’t abuse it. Number 7 is important for making sure any data being sent back and forth is not malicious in content. Number 8 is where we apply defense in depth strategies to everything so we can prevent attacks. Number 9 is important when we are testing the application to make sure it is ready to be used by users. Proper testing techniques like fuzzing and penetration testing are good practices. Lastly number 10 is where we are supposed to have good guidelines for preventing a coding fatality. Following the C++ standard library principle helps us to prevent issues in code specifically with C++. |
| **5** | For this next slide, we are going to go over each of the coding standards I found very important for Green Pace. I ranked each of them based on priority since I thought that there were some of these much more important to watch than others. For example, I put the policy about deadlocking almost last since not every application will use threads and put enumeration casting first on the list since it has to do with using values. |
| **6** | Encryption is important to keeping data safe. But there are 3 kinds of different types of encryptions to keep in mind such as, at rest, in flight, and in use. At rest means basically that the data is “resting” in a hard drive or storage. In flight means the data is being transmitted somewhere. In use means the data is being used by the user currently. Each of these are important to keep in mind when wanting to protect data since a hacker could easily intercept or hack a system and take a message containing confidential data and take it for personal gain. |
| **7** | So here I want to explain about Triple-A policies that should be put into practice. Authentication is the process of making sure a user is who they say they are. For example, requiring two-factor authentication is a 2nd way to make sure a user is the actual user and not an attacker. Authorization is where the system will match the right permissions with the right user type. This could be like matches admin permissions to the admin user such as reading, writing, and deleting files while not allowing for regular users to do these things in regular user permission. Finally, accounting is where the company accounts for every change that is happening in the system. So, making sure the application is logging who was accessing what at a certain time is one example of accounting. |
| **8** | For this slide, I will be explaining our unit testing process for the application and some examples of what I tested. Here is an image of the program before any unit tests were written. For the first test case, one of the issues was solved when I created a test to help see if a single value was being added to the collection correctly. Then, created one to test on 5 values. Next test case was verifying whether or not the capacity is greater than or equal to size for different numbers of entries. The 3rd test case was to verify if by increases entries if the collection will resize accordingly. The 4th test is for testing the collection to make sure it resizes accordingly as the entries decrease. The 5th test was for decreasing also but this time, I needed to test what would happen if the collection entries went to zero. Test case 6 is where I tested to see if the application is properly erasing with the clear() function from the collection. Test 7 was to test erase(begin,end) to see if the collection would erase from beginning to end. Test 8 is to verify whether or not the capacity is increased based on reserve and not size of the collection. Test 9 is for verifying that the out of range exception is being thrown with indexes that are out of bounds. Lastly, for test case 10, I had to make 2 different tests, 1 being positive and the other negative, on anything to do with the collection. At the very end, I commented out the “AlwaysFail()” function and the tests all passed. |
| **9** | For this slide, I am going to explain a bit about how DevSecOps is supposed to work and how important it is when implementing it into a project. Automation should be implemented at every stage of development, especially mitigating early issues instead of leaving every compiler warning to the end. Defense in depth should be utilized throughout the cycle as this will protect the program from attackers. The compiler should also be used throughout each stage of the chart since it is meant to help us to find problems too, but not every issue is solved with the compiler. Automation tools should be used throughout the development process to not leave any stone unturned when it comes to mitigating vulnerabilities and helps with finding hidden issues. CPPCheck is a really good automation tool for finding vulnerabilities that the compiler could not find. |
| **10** | The problem with waiting too long to mitigate something is that more harm is done. This is because time, money, and reputation are in the mix. The longer time something stays unfixed, the more time an attack can happen and be costly for the company. Acting in the “now” will not only save the company time, money, and their reputation, we will also save the data of thousands of customers. The solution is to have security measures in place throughout the project. Leaving it to the end will have consequences for the company. |
| **11** | I recommend always talking with your team if you’re confused on something or in general just to let them know what’s going on with the project. A lack of communication causes the most mistakes to take place. Second, please go over the coding standards that correspond with the programming language you will be using ahead of time and during usage. Third, implement defense in depth at every level even if it seems cumbersome to do. It will save so much time, money, and the company’s reputation. Fourth, testing throughout the development process rather than leaving it for the end is going to leave room for less mistakes. If we just test at the end, we waste time since we will most likely have to go back and change some stuff versus only changing some things while we test. Fifth, keep all accounts at a least privilege permission. We do not want anyone who isn’t supposed to have access to certain elements to have access. Lastly, implementing Triple-A practices will also help us to keep customer data safe and be another line of defense. |
| **12** | I want to conclude with a very good example of how not implementing standards and defense in depth practices can be harmful. Alibaba is a company that has many different holdings such as Aliexpress (shopping) and Alibaba Cloud (cloud services). In 2019, a couple of employees from a data mining company hacked into Alibaba and stole 1.1 billion customer data files. Though the attack was not used for personal gain, it still showed Alibaba did not put much effort into implementing defense measures, as they did not require usernames and passwords to access the data. This is why defense in depth measures along with Triple-A practices need to be always in place. Also, having knowledge on the hacker types (black, grey, and white hat) and what each’s motivations are can help prevent many attacks before they start. Knowing the enemy can help us to get ahead of the game before it even starts. |