**Heriberto Torres**

**6/20/2021**

# CS 405 Project Two Script

**Video URL:** [**https://youtu.be/\_hw9Bk1RFQg**](https://youtu.be/_hw9Bk1RFQg%20%20)

| **Slide Number** | **Narrative** |
| --- | --- |
| **1** | Hello, my name is Heriberto Torres and today I will be discussing the Security Policy developed at Green Pace. |
| **2** | Here at Green Pace, we have developed a security policy to standardize principles for our coding standards, encryption, and application of Triple-A policies. Each piece of subject matter adds additional layers of security to support the defense-in-depth best practice and eliminate potential vulnerabilities. |
| **3** | Using a threat matrix we have categorized different security threats. Likely threats include denial of service where a cyber-attack can flood a computer or network so that it is unable to respond and man in the middle attack where a hacker can come between a two-party transaction to steal data. Under priority we have SQL injection where a hacker can insert unauthorized SQL statements to interact with the database and Password Attacks where hackers steal or brute force user login information to gain unauthorized access. For low priority there is phishing in which a hacker attempts to trick users into giving up their login information. Lastly, under unlikely, we have malware which is malicious software often received by clicking on a suspicious link or attachment. |
| **4** | In order to support the development of software at Green Pace, we have developed ten core security principles. We must validate any input submitted by a user, heed compiler warnings to eliminate potential security flaws, design our software to implement and enforce security policies, keep our projects as simple as possible to mitigate vulnerabilities, default deny as the standard level of access, execute functionality with the minimal amount of privilege required, sanitize any data transferred between systems, utilize multiple layers of defense, utilize effective quality assurance techniques and procedures, and lastly, develop and apply a secure coding standard for the platform and language used. |
| **5** | Green Pace has also developed 10 Coding standards to follow. Developers must consider data types and code around their restrictions. Data values handle different levels of precision, so it is important to handle data conversions accordingly. Strings should contain sufficient space for character data and the null terminator. Developers should follow proper safer procedures when engaging with SQL to prevent SQL injection. Memory size for different data types should be in consideration as well as concern for memory size when changing data types. Assertions should be used to test assumptions rather than state the obvious. Exceptions should be used for exceptional situations and not as a means to keep a program running. When managing files, ensure proper file handling procedures. Ensure a value is returned from all potential paths of a function. Lastly, when handling containers, ensure that the indices and iterators are of a valid range. |
| **6** | The three encryption policies Green Pace will abide by are encryption at flight, encryption in rest, and encryption in use. To elaborate further, encryption at flight is the protection of data in transit such as utilization of VPNs or HTTPS. Encryption in rest is the protection of data being stored such as data found on disks, files, and databases. Encryption in use is the protection of data being used at a particular instance such as sign in attempts or charging a customer’s credit card. |
| **7** | For our Triple-A policies here at Green Pace, we support the implementation of authentication, authorization, and accounting. Authentication is used to identify and confirm who or what is accessing the application in order to restrict access or grant privileges as needed. Authorization is the method of granting specific users privileges or restrictions based on their level of access to ensure specific data or functionality is only used by select individuals. Accounting is the implementation to measure a user’s time spent accessing the application, when and where it is accessed, and how often it is accessed. |
| **8** | For some examples of unit tests performed here at Green Pace, we have our first test of a character array. This function will be successful if the user inputs 20 characters or less. However, it will fail if the user exceeds the character limit of 20. This could be fixed by utilizing strings to accept user input. |
| **9** | In this second example we test return points. This function will run successfully if the value passed in is less than 0. However, if a value greater than or equal to 0 is passed in, the function will throw an error. This would be fixed by ensuring all function paths return a value. |
| **10** | For the third example we test container size. The function will succeed if the number passed in is greater than or equal to 0; it will fail if given a negative value. This could be fixed by only accepting arguments that are positive. |
| **11** | For the fourth and final example, the limit of data size is tested. This function will succeed if it is passed a set of numbers that will not exceed the limit when incremented. It will fail if the numbers passed will exceed the data type limit when incremented. A fix to this would be the create a temp variable to check and ensure the incremented value is greater than the previous value. |
| **12** | In order to automate enforcements of the standards of this policy, it would be beneficial to start with security in mind. How code is implemented and unit tests are made start with this policy in mind. By doing so, time and effort would not get wasted into adding security features that could even change the design of the application after much of the work has been done. No code should be committed to the main branch of the application until formally reviewed following this policy as a guideline. In addition, it would be important to review policy standards before any work reaches production as well as before pre-production begins to uphold security standards and catch any mistakes that may have slipped through. |
| **13** | The dev-ops pipeline is a means to utilize automation and tools that allow developers and operations professionals a means to work together and streamline the process of development, deployment, and operations. This involves the act of continuous integration, deployment, feedback, testing, and monitoring and operations. Within this process, the testing and monitoring stages would contain security automation to run the compiler, run any unit tests built, and continuously monitor the health of the application as well as scan for potential hazards. In the previous diagram, these tools and automation would fall under the verify and test category, the transition and health check category, and the monitor and detect category. |
| **14** | When it comes to security, it is important to not leave security to the end. Some security concerns can influence the structure of an application such as user roles and permissions. Leaving such considerations until the end can lead to far more time, work, and effort required in order to redesign the application to utilize these security features. Waiting to consider security can also lead to the gathering of unnecessary data which could waste time, space, and include sensitive data that may not be required for the application.  Steps that can be taken to prevent such threats is to design projects with security in mind first. This can be something simple such as collecting sensitive information only if it is necessary or a more technological approach such as user authentication, user roles, user permissions, limiting access time, enforcing strong passwords, proper password storage, monitoring network activity, and much more. |
| **15** | Recommendations to further improve this security policy would be the addition of more coding standards as well as additional coding examples for each existing and future coding standard. A section about cyber awareness training for employees would also help secure against threats such as phishing and malware that could come from any employee who is careless. Furthermore, it would be beneficial to list common hacking methods that are used and how to protect against them. |
| **16** | Additional standards that should be adopted to prevent future problem are adding in standards to handle common object-oriented programming concerns such as the usage of public and private variables. A standard on how errors are expected to be handled. How and why to use environmental variables. Lastly, to set a standard of data sanitization of any unpredictable source. |
| **17** | This concludes my presentation on the Green Pace security policy. Thank you for watching and have a wonderful day. |