# CS 405 Project Two Script

**Benjamin Abbott**

**December 9, 2022**

**Project Two: Security Policy Presentation  
  
Link to presentation: https://youtu.be/KFqChesMX7s**

| **Slide Number** | **Narrative** |
| --- | --- |
| **1** | Hello, my name is Benjamin Abbott and I would like to Welcome to my Security Policy presentation for Green Pace. |
| **2** | In this presentation we will go over how defense in depth will be used to layer different security measures to ensure that security is a strength and focus and how this security policy will be used to provide consistent approaches and methodologies to implement secure principles to the software development and provide levels of security to Green Pace. |
| **3** | As you can see here this is a threats matrix divided into 4 quadrants. Threats come in different levels and need to be categorized in priority and in likelihood. Threats that are likely and of high priority are the most dangerous to a system and will need to be handled urgently, while threats that are unlikely and of low priority can wait until nothing of a higher priority is left to be handled. |
| **4** | The ten core security principles are:   * Validate input data * Heed compiler warnings * Architect and design for security policies * Keep it simple * Default deny * Adhere to the principle of least privilege * Sanitize data sent to other systems * Practice defense in depth * Use effective quality assurance techniques * And Adopt a secure coding standard.   This list shows where my coding standards fall within these principles. |
| **5** | These coding standards are important for Green Pace to implement during development. They are:   * Do not access freed memory * Prevent SQL injection * Do not attempt to modify string literals * Honor exception specifications * Never use assertions to validate method arguments * Allocate sufficient memory for an object * Include the appropriate type information in function declarators. * Do not begin integer constants with 0 when specifying a decimal value * Do not leak memory * Use a static assertion to test the value of a constant expression   Implementing these standards will ensure a strong foundation for secure code. |
| **6** | Encryption is also an extremely important aspect to any system or application. There are three types of encryption and each is as important as the next. They are:   Encryption in rest refers to data being encrypted while being stored. This data is protected by a key so that only those with the key can access the data. This will help protect data from being accessed by malicious actors or prevent people who do not have proper access from accessing this data.  Encryption at flight refers to data that is encrypted while being transmitted. Before being transmitted this data is encrypted so that those who try to intercept it in transit are unable to access the data. The intended recipient will have the key to the data so that it will be unencrypted when reaching the destination.  Encryption in use refers to data that is encrypted while being used. This is used when security levels are instated, and employees are only granted access to data that is a part of their level of security access. |
| **7** | The triple A framework is another important security measure that provides instructions on intelligently controlling access to system resources. They are:  Authentication refers to the process of authenticating the user. This is done by implementing usernames, passwords, as well as two-step verification that adds confirmation emails or texting codes to users to verify identity.  Authorization refers to the level of access that the user has and authorizes the user access to certain data.  Accounting refers to the record keeping of what users do while accessing the system. It records the date, time, user, and what was changed or accessed by them. |
| **8** | Unit testing is a valuable tool that will allow developers to test the smallest pieces of code that can be isolated in a system. This is done during the programming phase. The following four slides show how unit testing can help detect different vulnerabilities. |
| **9** | This test is a Capacity unit test. This is a positive unit test that uses assertions to verify that the capacity is greater than or equal to a variety of vector sizes.  If larger capacity is needed it is important to test for higher capacities before implementing. |
| **10** | On the next slide we have a reset size to zero unit test. This is a positive unit test that uses assertions to verify that the vector is empty when resizing to zero.  Resizing a vector to zero can be very useful. Don’t use this if you are trying to delete elements. Instead use clear or erase functions. |
| **11** | The next unit test is an Out of range unit test. This is a negative unit test that verifies that an out of range exception is thrown when calling at function for an element out of bounds.  It might be useful to test the out of range with other functions as well, such as swap or push\_back functions. |
| **12** | The last example of a unit test is a length unit test. This is also a negative unit test that verifies that length exception is thrown when a vector is resized over the allowed max size.  It is important to know your boundaries with different classes. Be sure to test other containers if using them. |
| **13** | Automation is a vital part of security and detecting vulnerabilities. The figure you see here is the DevSecOps pipeline. |
| **14** | The DevSecOps pipeline is a way to look at the software development lifecycle with the incorporation and prioritization of proper security techniques and protocols.  External tools are used throughout the DevSecOps pipeline to complete tasks such as scanning for vulnerabilities, running analytics, monitoring and detecting events and intrusions, automated response services, and for many other critical needs. |
| **15** | While there are many benefits to following the DevSecOps pipeline and developing with this level of security in mind, I believe that all the short term risks taken will be far outweighed by the long term benefits that it provides and in the end I believe that it will save both time and money, while producing stronger and more secure systems and applications here at Green Pace. |
| **16** | Some recommendations that I have explained in this presentation and believe that will help Green Pace produce the highest quality code are:   * Create standards for future development. * Develop with DevSecOps pipeline. * Use encryption for all data in flight, at rest, and in use. * Use the Triple-A framework strategy * Use unit testing * Use external tools |
| **17** | In conclusion, While it is true that there is no such thing as 100% secure code. It is vital to ensure that you have developed the most secure code that you can. Implementing Defense in Depth and using the DevSecOps pipeline along with adopting the standards and tools discussed today during development, will not only improve the security of the systems that will be developed, but it will also lead to consistency and ensure that only the highest quality of software is produced at Green Pace. |