# CS 405 Project Two Script

# Jamilyn Glidewell

# 4/19/2022

Complete this template by replacing the bracketed text with the relevant information.

| **Slide Number** | **Narrative** |
| --- | --- |
| **1** | Hi everyone! My name is Jamilyn Glidewell. I am here with you today to discuss the new security policy that we have created for Green Pace’s developers and IT team. |
| **2** | This presentation will discuss how we are building defense in depth. Defense in depth is a concept in IT security that builds multiple layers of security between our systems, code, and data and those outside the company that would like to steal that information. |
| **3** | Potential issues are categorized in a matrix. In this matrix we have:   * Likely – this means an issue is more probable to occur. * Priority – this means an issue is important for us to consider for prevention. * Low Priority – this means that even if an issue happens, it is less critical. * Unlikely – this means that an issue probably wouldn’t happen. |
| **4** | The 10 principles of security are the basis for the entire policy. These 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 * Adopt a secure coding standard   While these principles don’t guarantee a secure system, if we implement all 10, along with some other best practices, we will be as secure as we could hope to be. |
| **5** | The threat matrix is expanded here with some specific rules and what those standards mean. Because there are quite a few, I will just highlight one or two.  First, not writing ambiguous declarations helps ensure the code can only be understood one way, which is the way you intend it to be seen when you wrote it.  Also, not modifying standard namespaces helps avoid undefined behavior if they are not used correctly. |
| **6** | There are few key places to encrypt data to make sure it remains secure throughout all the functions that might happen on that data. Almost all data always needs some type of encryption to protect it, and those types of encryptions vary based on what is happening to the data.   * Encryption in rest – this secures the data as it is written to storage and decrypts it as it is pulled. By using a key, unauthorized data can be prevented from being written to the systems as long as they don’t have the key. * Encryption at flight – this secures the data as it moves throughout the systems. The type of data and the way the data is being sent changes the way you have to encrypt the data. This is important, though, because it is the place that data is likely to be intercepted. * Encryption in use – secures data as it is used in memory. This prevents data from being leveraged someone is able to execute against memory. |
| **7** | The “Triple-A” policies all sound similar before they are defined. There can be quite a bit of confusion around the differences in the three, so we will define those here.   * Authentication – makes the user prove who they are. This is commonly done through usernames and passwords * Authorization – once a user has proven they belong in the system through authentication, they are able to access the data they need to successfully do their jobs, but no more. This is the concept of least privilege. * Accounting – keeps track of who did what and why. This sounds like the company is tracking people, but really what we are doing is making sure that there isn’t some strange behavior by a particular user that might indicate a problem. This also helps us to continuously review the data that is being used to make sure it still makes sense for our purposes. |
| **8** | Unit testing is an important part of secure development. Because unit tests are almost always done to make sure the code is functioning the way it is supposed to, it is smart to include security-based unit tests as well, to identify any potential weak points in the code. Some examples follow on the next slides. |
| **9** | One example of a unit test would be testing for strings. You could run tests to validate if there are strings in the code, because there almost always will be somewhere. Once you’ve confirmed those strings, you will validate that those strings are doing what they are supposed to do, no less, no more. |
| **10** | Where possible, and where companies have investment in it, code should be tested through automated tests. This helps to make sure there are consistent tests run across the code base for a baseline of security and functionality. |
| **11** | DevSecOps is an expansion of DevOps that includes a security component. This process uses development, testing, security, and operations together to help build, test, and deploy code securely and quickly.  There are many tools that companies use for DevSecOps. Tools like Git, Jenkins, Selenium, and others help build the suites that make continuous testing and deployment possible. It is important that each of these tools is secured and any data transferred between them is also secured. |
| **12** | Risk of this policy is that it takes a little more time to design and build code with security involved. It also means that developers have to have a security-first mindset, which might be a bit different than what they are used to.  However, the most important part of the security policy is also the biggest benefit. The fact that code is more secure and that security issues are addressed before they even come up helps to keep the company and the company’s customers more secure. |
| **13** | The biggest risk with this proposal is that existing developers likely don’t think security-first right now. Having to change the way the development teams think about development will be a challenge.  However, it is worth the extra training and effort to get the teams up to speed. It is very important that this policy be implemented as soon as possible to help increase the security profile of the company as a whole. |
| **14** | In conclusion, it is very important that we adopt the entire security policy in full. Providing training to our team members will help ensure that any gaps in understanding between the policy and current practices are fixed. Also, there should be regular audits of the process to keep security at the front of everyone’s minds. |