# CS 405 Project Two Script Template

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

Name: Kelly Lewis

Date: 6/25/2023

Class: CS-405

School: Southern New Hampshire University

Video Link: https://youtu.be/sTmH0VSqjGg

| **Slide Number** | **Narrative** |
| --- | --- |
| **1** | Hello, my name is Kelly and I work at Green Pace as a software security analyst.  I have been tasked with creating a new security policy to be applied to our projects here at Green Pace. |
| **2** | The overview for this presentation is the application of a Defense in Depth.  To start, Defense in Depth is a layered defensive approach to software security that encompasses software development and deployment. Aspects of a layered defense encompass both physical and non-physical attack vectors.  This presentation is targeted at the non-physical aspect of software, or rather the development of software.  Defense in Depth is a multi-layered deployment of numerous defensive layers. Each layer is chosen to cover potential exploits or holes in previous layers. The overall goal is to create a solid defensive bubble that can cover as many potential attack vectors as possible.  The focus of the security policy and this presentation is to show how focusing on security, at a developmental level, plays a crucial part in this encompassing defensive strategy. |
| **3** | Threats are classified by several factors. The factors being presented here are focused on likelihood and priority.  While a threat may be categorized as unlikely, the severity can be higher with higher remediation costs overall.  No matter the priority or likelihood, they are all important to recognize as you work.  The ranked system is designed to help guide how to prioritize or where to emphasize potential issues. |
| **4** | In the security policy documentation, ten principles have been lined out for software security and development.  Each principle, while not shown here, can be mapped out to specific standards.  The missing standards from the Default Deny, Adhere to the Principle of Least Privilege, and Sanitize Data Sent to Other Systems are principles that apply to a multitude of potential situations. More on this later in the presentation.  For each principle that directly coincides with a current standard, the standard is listed in the documentation.  Each of these will go into effect and should be reviewed more in the documentation. |
| **5** | The ten coding standards that have been selected are shown here and can be referenced further in the documentation.  Currently, the order here is not how it is ordered in the documentation as this list is ordered by the ranking system mentioned earlier. This list is ordered based on the severity first, followed by the likelihood, then by the remediation cost.  Looking at the table, the coding standard is based on the attack method or common areas that lack security focus and should be given some priority.  The next column is the label, this is broken down by STD or the standard, followed by the number associated to the standard in the documentation. Finally, the language that the standard references.  The final column is the name of the standard. As can be inferred, each name references the type of vulnerability being shown. On the previous slide, most of the principles had these standards listed with them. |
| **6** | Next up, we will be talking a bit about the new encryption policies and what they are referring to.  Data exists in three states, at rest, in flight, and in use. This policy covers that no matter which state the data exists in that it should have encryption.  Data in rest, refers to data that is not being transferred or in use. Typically this is stored data.  Data in flight, refers to data that is being transferred. This refers to e-mails, file transfers, or when you are accessing remote devices as packets are transmitted from host to client and vice versa.  Data in use, refers to data that is actively being accessed. This is typically files that is being modified or viewed. |
| **7** | Triple A. What is Triple A? Triple A is Authentication, Authorization, and Accounting. How does this relate to security or software development?  Authentication should be used in our development and utilized in our projects. Authentication verifies a user and confirms the user’s identity.  Authorization is the policy of creating and using system roles to only allow users to access what they need to access.  Finally, Accounting is the policy of logging user access, changes, and other actions a user takes while accessing the system.  These can be crucial in preventing, halting, or investigating possible attacks or breaches. |
| **8** | The following slides are examples of unit testing and the results of each test.  Each of the chosen tests can be used to check for issues pertaining to data manipulation.  The first test is checking if a resized collection vector is resized to prevent an out of bounds exception or error. |
| **9** | The next unit test checks to ensure that A) an exception is thrown and B) that the thrown exception is an out-of-range exception. This is useful when ensuring that exceptions are handled properly, and that error codes are assigned properly. |
| **10** | The next unit test is designed to ensure that values set in two locations are not equal if a comparison is made. This can be useful in database searching and ensuring that set values are not equal, if they truly are not equal. |
| **11** | The final unit test checks to ensure that when a value is set at a specific location, that the value is actually set. Again this can apply to many situations, for example database manipulation verification. |
| **12** | This is an infographic of the DevSecOps Software Development Lifecycle.  There are many points on this flowing diagram that show where software security testing can come into play. Automation can assist in the application of checking for security vulnerabilities. |
| **13** | First, what is DevSecOps? DevSecOps is a modification of the previous DevOps or development operations that now includes security. The previous diagram shows how security is now included in the Software Development Lifecycle or SDLC.  Tools. These are security frameworks or applications that can be used for testing and monitoring security throughout the development process. Many of these tools are designed to automate the testing and checking of software or code for security vulnerabilities. Many of these tools will be deployed throughout the SDLC. |
| **14** | Risks and Benefits. The risks and benefits of acting now and instilling a security first mindset or waiting, and saving security for the end.  Acting now has a plethora of advantages that include, but are not limited to: mitigating costly errors or issues, increasing overall software security, minimizing vulnerabilities, etc. All for a marginal bump in cost, both time and money, yet returns maximum yields in the area of secure software design that can mitigate extremely costly data breaches and other issues.  Waiting on security can save some cost, again both in time and money, and leave resources focused on feature creation or expansion. Yet, this increases potential security risks and allow for more vulnerabilities upon release. These can and have led to costly security breaches.  Ultimately the philosophy of “Don’t leave security to the end.” Is important on many fronts and can help to avoid costly issues. |
| **15** | Further recommendations.  I base my current further recommendations on the lack of direct standards for reference in the following: Default Deny, Adherence to the principle of least privilege, and sanitization of data sent to other systems.  Default deny is the approach of coding to deny access by default. This falls in line with Authentication from Triple A.  Adherence to the principle of least privilege again refers to the authorization policy in Triple A.  Sanitization of data sent to other systems, this is about ensuring that no harmful data or injected code is transmitted from one device to another.  Developers would benefit from a direct standard for each of these. |
| **16** | In conclusion, security should be at the forefront of modern software development. Don’t leave security to the end.  SEI CERT security recommendations, rules, and examples are there to help developers create secure and sound code. Use the reference and documentation.  As new features are adopted into the project, security should be re-addressed.  Add new standards to cover the listed principles of default deny, adhere to the principle of least privilege, and sanitize data sent to other systems.  Thank you for watching and have a good day. |