# CS 405 Project Two Script Template

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

| **Slide Number** | **Narrative** |
| --- | --- |
| **1** | Hi, my name Ernest Enriquez and today I will be presenting you my security policy. The information you are about to see is based on a thorough analysis of secure coding principles but also our current infrastructure. |
| **2** | Let’s talk about defense in depth. This security policy is designed to provide an end-to-end solution of system checks we should implement. It is broken down into different sections that describe unit test, risks, and benefits. We will also cover the 10 principles of secure coding. |
| **3** | Our solution is at risks from a variety of threats. Some risks and vulnerabilities are more important than others. For likely risks, we have standards 1,2,3,5,6,9, and 10 and for unlikely we have 4, 7, and 8. Our top priorities are 2,3,4,5,8, and 9. And the low priorities are 1,6,7, and 10. Some things to consider when making decisions here are how much it costs to mitigate issue and recovery. |
| **4** | In this section we’ll cover the 10 principles to coding securely and then we’ll apply them to our coding standards. When consideration of principle we think about where they fall in priority. Validate Input data we have standards 4 & 8. I applied the same logic to the other as well, which you can see in our list. |
| **5** | The ordering of coding standards is based on priority. To mitigate potential harm to sensitive data collected and exploitation of the damage. Practices that utilize sql injection should be prioritize as highly likely. These are actions which ensure we are getting valid types and members. |
| **6** | Finally, we have encryption in use which is essentially having the data encrypted until you authenticate a user has access to said data. This policy applies because like before if we open up one of the DiD components, we make the whole system potentially vulnerable. This is one of the most important in my eyes because it could be a user-facing application and if data isn’t properly stored, you could potentially release unencrypted data to a non-malicious user. Encrypting data until it is needed is a fantastic idea and a policy we need to uphold.   * Encryption at rest - Data-at-rest encryption is a method of protecting the local storage of data through encryption. The purpose of data-at-rest encryption is to protect application data from attackers who have access to the storage device but not the application itself, making the data * physically unreadable. This policy applies because every state must protect data in order to have a truly in-depth protection strategy * Encryption at flight is essentially just encryption in transit. As the data is being transmitted, the data should be sent through encrypted channels. One example would be using HTTPS. This policy applies because where we already have our data encrypted in rest, why would we decrypt the data to transmit it, opening ourselves up to attack and breaking down our defense in depth strategy? We shouldn`t, which is why we need encryption at flight * As before, this policy is enforced because exposing one of the defense-in-depth components could potentially make the entire system vulnerable. This is one of the most important things in my opinion as it can be a user-centric application and if the data is not stored properly, it can potentially pass unencrypted data to a non-malicious user. Encrypting data until needed is a fantastic idea and a policy we should support. |
| **7** | The authentication process is usually done through a user login which we are all aware with because we use them daily. This is a process used to gain access to a network. We validate that each user must also have a unique set of credentials with assigned user access.  Next, we have authorization. This deals with the level of access and permissions account users will have. It’s important to be able to identify who access to what data.  Finally, we have accounting. This is the step we keep track of how much bandwidth each user utilizes when connected to the network. This is a fantastic last line of defense to help secure our network. Even if an attacker manages to get beyond all our other security procedures, we'll now be able to see what they're up to on our network |
| **8** | For each of my unit tests, it is easy to see how beneficial they can be to address the different security concerns that we are presented with. I have provided the first, two negative testing examples.  First, we have a non-compliant code where our containers are subject to vulnerabilities. Here we use CTR52-CPP to ensure that the library function does not allow for overflows. |
| **9** | In the non-compliant code, they can result in an exception being thrown. Coding Standard ER50-CPP avoids programs from terminating due to errors. A terminating application can be critical as it can potentially return unsecure data to the end user. |
| **10** | In this info graph we have a summary of the automation lifecycle. This describes the pre-production and production life cycles. Our infographic here will help us get a better understanding of the planning process. |
| **11** | The Green Pace DevOps process can be automated for enforcement during the Transition and Health check phases of the cycle. This is the stage of the process in which newly verified and tested settings are deployed. The deployment of these security settings can be automated by pushing updates to all live systems that include a set of pre-configured rules and standards. This will reduce the possibility of human error when configuring these rules and standards manually. Normally, penetration testing is done manually; however, a penetration testing lab could be automated to test newly released settings on a continuous basis. If this automated system gains unauthorized access to the new code, an event will be triggered immediately. Finally, the system would begin by closing all necessary ports and blocking IP addresses. |
| **12** | So, what are the risks vs benefits analysis when implementing this security policy? Some risks include to much implemented security could slow down development. It could also make the design of our application difficult and costly. However, there are benefits which include a more secure product for customers. Also a reduction of potential security threats later on in development. |
| **13** | My recommendation to green pace. Adapt to policy changes, change the scope of the project, and additional unit testing. |
| **14** | In conclusion we described our security policy in depth. Some key take aways I want to end with though to recap my security policy’s top priorities. Data Sanitation must implement to ensure secure input. We also must develop and implement encryption policies to protect vulnerable data in the event of an attack. Our team must constantly test for vulnerabilities. Lastly, ensure we practice the 10 principles of secure coding. |