# CS 405 Project Two Script Template

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

| **Slide Number** | **Narrative** |
| --- | --- |
| **1** | Hello and welcome, my name is Laura McAroy and I will be walking you through this presentation on security policies today. |
| **2** | Defense in depth is the gold standard for security. As you can see from the illustration, it involves adding several layers of security, covering all aspects of a system. This way, when one layer of security is broken, there are still several layers in place to help keep the system secure. |
| **3** | This is the threat matrix, sorting our 10 standards into four categories. You can easily see the correlation that most of the threats that are likely are also high priority, while most of the low priority threats are unlikely. We will go into these standards and priorities further in a minute. |
| **4** | These are the 10 principles to consider when designing secure software, and you can see on this slide where our 10 standards fall on this list. Most of our standards fall under 3 categories, Validating Input Data, Architect and Design for Security Policies, and Adopt a Secure Coding Standard. While not discounting the other principles, it can be argued that these three are priority when developing secure software. Proper validation of input data reduces tremendous amounts of vulnerabilities. By designing the structures of your code with security policies in mind, and implementing an architecture that enhances security in the initial design and development phases, you save a lot of time and money going back and fixing vulnerabilities found later on. In doing this, you adopt a secure coding standard in the beginning phases and make security a priority throughout the SDLC, therefore adopting a secure coding standard. |
| **5** | For my list of coding standards, I have grouped them from highest likelihood and priority to lowest. The first four standards are the ones that landed in both the “likely” and “priority” categories of the threat matrix. All four of these principles have a priority rating of 18, which means that they are both likely to occur and are of a high severity.  The next group contains standards that are either likely or high priority, but not both. Standard 3 being a high priority but not likely, and standards 2 and 9 being very likely, but not high priority.  Standard 7 is next in the ranking, as it is probable to occur but listed as a low priority vulnerability, followed by standards 1 and 6, which are both low priority and unlikely to occur. |
| **6** | Encryption Policies  Encryption at Rest is the process of translating saved data to another form of data via encryption. It allows for the protection of sensitive data and requires authorization to decrypt the information.  Encryption at Flight is the process of encrypting data that is moving over a network. This policy is especially important when moving data over the internet. It protects sensitive data from being breached during a data transfer process.  And Encryption in Use is the policy where data is never left unsecured regardless of its lifecycle stage. Data access is monitored and controlled through authorization during this approach. Data requests are analyzed and responded to in real time allowing awareness of suspicious activity. |
| **7** | Triple-A Policies  The authentication process is where users are confirmed through proper credentials such as usernames, passwords, two-factor, or biometrics, before they can gain access to a system. This policy is especially effective in controlling unauthorized access to use accounts.  The authorization process is where levels of access are determined for system users. Depending on what type of user is accessing the system, limitations to access can be established appropriately, think principle of least privilege. This approach would enable only those with admin privileges to gain access to databases and employee records, make changes to those records, and add new users to the system.  The accounting process is where system users are monitored and logged according to user interaction. This approach allows administrators to keep record of when data has been accessed or changed in the system and who made the changes.  These processes all work together to give a system a multi-layer defense. |
| **8** | These are some examples of Unit testing. This first unit test here ensures that an exception is thrown when an index is out of bounds. A similar test can be written and applied for standard 1, do not cast to an out-range enumeration value. |
| **9 - 10** | The next two unit tests are both ways to ensure that the data in the collection has been erased. If data is no longer needed within a system, it should be erased to prevent memory leaks and errors. |
| **11** | The final unit test example is ensuring the size of the collection. This coincides with standard 10, guarantee that library functions to not overflow. While there are more efficient ways to ensure that your functions do not cause buffer overflow, it is almost important to test that your database is the size you need it to be. |
| **12** | Here is an illustration of the DevSecOps pipeline. What I would like to do is add automated testing to the verify and test phase. Implementing automated testing will allow developers to test their code more efficiently with minimal interaction, while also holding each developer’s code to the same standard and testing. There are several tools available to help throughout this process. |
| **13** | SD elements can be utilized in the initial assess and plan and design phases. This tool automates security requirements based on company needs and technology.  IriusRisk is the next tool that can be used throughout all phases of the SDLC. It applies standards such as OWASP and creates threat models and manages security risks.  Parasoft Tool Suite and Chef Inspec can be used during the Verify and Testing phase to run automated software testing. Chef Inspec compares the software’s current state to the desired state and generates a report based off of it’s findings.  Splunk and Tripwire can both be utilized during the monitor and detect phase. Splunk manages logs and Tripwire is a monitoring tool for configurations, ensuring they comply with both internal and external policies. |
| **14** | Don’t leave security till the end. By adopting a secure coding standard in the beginning, you keep security a priority through the entire lifecycle of the system. Implementing security measures early on minimizes the amount of time and money spent correcting vulnerabilities found in later phases of development, which increases the speed and efficiency of the development process.  There are some risks involved. The first being that this requires a significant shift in mindset to become security focused. In order for this process to be successful, you will also have to ensure that your team has enough knowledge to implement security measures and utilize tools correctly.  That is why the first step to be taken needs to be educating and training employees on security policies and current standards. Training on tools and processes should be included as well, so that they can be implemented and used successfully. |
| **15** | My first recommendation would be to assess our current policies and compare them to the policies listed here, along with other current standards and best practices. From there, we can discover gaps in our policies and prioritize them so that we can begin to develop a plan to correct them. Education and training is absolutely necessary for everyone involved in the development process. Automated testing and monitoring tools like the ones listed earlier need to be explored further to determine which one best meets our needs, so that they can be utilized. And a system needs to be put in to place to ensure that our new policies are being used. I have listed a few recommendations here. |
| **16** | In conclusion, we first need to ensure our policies match current best practices. We then need to adopt a secure coding standard, develop our software with security in mind. We need to implement a defense in depth approach, create multiple layers of defense against attacks. We need to incorporate automated testing into our testing phases, ensure our staff is formally trained in implementing security measures, communicate any new policy changes, and determine which security tools will work best with our system, so that we can work more efficiently.  Thank you for your time. |