# CS 405 Project Two Script Template

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

| **Slide Number** | **Narrative** |
| --- | --- |
| **1** | This demonstration will illustrate Green Pace’s security policies, and how we are upholding them to the highest possible extent. |
| **2** | Input Validation is a powerful vulnerability mitigation technique, but should not be used alone. It is an integral part of any Defense In Depth strategy, because of its simplicity and its ability of preventing so many types of dangerous attacks. The illustration shown here depicts the numerous layers that go into it. |
| **3** | The likelihood of a standard not being met equates to the chances of a vulnerability being created. The priority of it is how severe the vulnerability potentially becomes. The threat matrix shown here illustrates how the likelihood and priority of a vulnerability coincide with each other. |
| **4** | The ten coding principles shown here are in order of necessity. Notice how Input Validation is the most important one shown here. This is because limiting what a user can input can greatly decrease the options hackers have at bypassing security. |
| **5** | The coding standards listed here are my choice for the most effective practices in upholding the coding principles on the previous slide. For example, preventing SQL Injection is one way to validate user input. |
| **6** | * Encryption in rest secures stored data. This mainly includes where and how the data is left unattended, such as within a database. Various encryption tools can protect data at rest, but various security policies (such as defense in depth) can greatly aid in protecting it. * Encryption at flight secures moving data. This may include data moving from one database to the next, between devices, etc. Various security tools can assist with this, but the main policy to be aware of here is to Sanitize Data Sent to Other Systems. This ensures that both the data and the new system is as secure as possible. * Encryption in use secures data as it is being analyzed, added to, removed from, etc. Securing data in use is best done by utilizing proper security protocols in regards to authorization and authentication. A great example here is the principle of least privilege, where each user is only given the minimum level of access they need to complete their own job. |
| **7** | * Authentication is the process of confirming if someone is who they say they are. This is most often done via a username and password. Though this does not make it impossible for a malicious user to access the data by any means, it is one piece of the defense in depth coverage, and can prevent numerous types of attacks. * Authorization is the process of determining and specifying the level of access each user is allowed. This is commonly done via the Principle of Least Privilege, which dictates that each user is only given the level of access that is necessary to complete their specific job, and nothing more. * Accounting is the process of recording and tracking who used the system, as well as when and how. Though this doesn’t necessarily prevent attacks, it can make it much easier to locate the source of attacks, figure out how the attacker bypassed the security, and patch it immediately. |
| **8** | Unit Testing can be used to identify vulnerabilities within a program. Here, Input Validation is repeatedly tested to ensure that the user is not able to perform any actions that will crash or confuse the program, such as attempting to add to an  already filled vector. |
| **9** | The image shown here depicts the process of developing software in the most secure possible method. Notice how the “Design” and “Build” sections on the left specify the need for secure methodology. |
| **10** | The DevSecOps pipeline is an automated method of producing secure software via the Software Development Life Cycle. This security integration minimizes the likelihood of an attack because the vulnerabilities of the system are at a minimum.  Though each of the coding standards and policies require different external tools, Astrée seems to be the most widely applicable. It’s a static code analyzer that checks for runtime errors and inconsistent behavior. It’s known as a thorough and trustworthy tool, which is why it’s recommended for almost every single one of these standards and policies. |
| **11** | The biggest risk of acting now (and by acting now we mean involving security policies in the development stages) is that it’s possible that a new type of vulnerability could arise if we later decide to increase the usage of the system. However, this is hardly a risk, because the whole point of acting now is that we act throughout the SDLC and beyond, so even if a new vulnerability presents itself, we will be ready to patch it immediately. The risk to waiting until the end of the SDLC to implement security protocols is that we could develop the program in an unsecure way, such as not properly validating user input from the start.  The benefit of starting now is that by the time the SDLC is “finished”, we have the most secure product possible. Though unit tests and various other types of checks are still necessary, we know that as many security concerns as possible have been properly mitigated. |
| **12** | Applying security policies throughout the SDLC is the safest way to develop a program, but it does leave a few gaps that require special treatment to be filled.    These gaps mostly revolve around rarer forms of attacks that general security policies don’t always catch. To properly fill these gaps, it is recommended that Software Developers be routinely trained on new forms of cyberattacks. This will help them identify and protect against these new types of vulnerabilities that general secure coding policies might not catch. |
| **13** | Cyberattacks are here to stay, so it is imperative that we make a constant effort at mitigating vulnerabilities within a system. Hackers are clever, and are always looking for ways to bypass security. To prevent them from being able to do so, we have to double down on the security practices that we already know, especially Defense in Depth, but we must also always be keeping up on the new strategies they are employing. Only be utilizing BOTH old and new security strategies can we keep our systems as secure as possible. |