# CS 405 Project Two Script Template

| **Slide Number** | **Narrative** |
| --- | --- |
| **1** | This will be a slide-show presentation on the security policy I developed for the company Green Pace |
| **2** | This security policy is designed to outline some of the common issues and vulnerabilities we face, as well as how to mitigate them. Using the concept of Defense-in-Depth, we can increase the level of security not only of the programs we develop, but also the company. With overlapping layers of security, the overall security will be greater than the sum of the individual components. |
| **3** | This slide gives a list of the 10 Principles of DevSecOps and the subsequent related standards that will be talked about shortly.  Validating input helps to ensure the safety of the system from attacks like SQL Injection or Brute Force attacks.  Listening to the compiler about the different warnings will aid in creating sound and secure code, in that it acts almost like a spell check.  Designing the program/software to be secure from the beginning allows us to mitigate potential vulnerabilities before they become a problem.  Keeping things simple reduces the overall development time and subsequently decreases maintenance time.  Default deny means that if there is any doubt, do not allow; this ensures safety at the slight cost of some user frustration.  The principle of least privilege means that only the amount of access needed to complete tasks will be given, to protect the system from unauthorized alterations/exploitations.  Sanitizing data being sent prevents unwanted eyes from viewing potentially sensitive material.  Practice DiD (Defense in Depth) to help bolster all aspects of the DevSecOps cycle.  Adopting a secure coding standard increases the repeatability and scalability of our programs in that everyone codes to the same standard and anyone on the team can pick up where someone left off. |
| **4** | The threats matrix gives a quick glimpse at how the subsequent chart lists out the different DiD standards. |
| **5** | On this slide, we have the 10 different standards that will be integrated into the Green Pace DevSecOps cycle. These are listed in numerical order with the different designations regarding their severity, likelihood, cost, priority, and level. The CPP and CLG denote which language the standard is referring to, with CPP meaning C++ and CLG meaning C. |
| **6** | This shows us the same standards, but in the order of what I believe to be the most detrimental if not followed.  I ranked the different standards based on severity first, remediation cost second, and the likelihood being third.  For a complete explanation of the different standards, examples, and mitigation tools, please refer to the attached DevSecOps policy document. |
| **7** | What is encryption?  Encryption is an exquisite way to help secure data in its different states through the use of an algorithm to translate the data from one form to another. Furthermore, the data can only be deciphered if the user has the associated key, which helps to protect the data no matter the state that it is in.  Encryption in rest is securing stored data by converting the data through the aforementioned means. This protects the information from letting hackers easily view the information that would be stored on the different servers/storage banks.  Encryption at flight is encrypting the data through the same means as in rest, but the data is being transmitted from one location to another. This helps to secure the data when it is in a vulnerable transit state.  Encryption in use is the idea of securing the data/information despite whatever state of motion the data is in. If a developer pulled data to run tests for a certain amount of time and has a local copy/accessing the server’s copy, then the data will be encrypted at all times to prevent potential leaks and breaches. |
| **8** | Another part of the DevSecOps program we will be instituting will be the inclusion of Triple-A Policies. Triple-A Policies are polices that deal with authentication, authorization, and accounting for the company.  What we mean by authentication is providing a method for identifying users based on each user having a unique set of login credentials to access a network. This can also be further bolstered through the use of two-factor authentication systems (2FA).  Authorization is giving people certain levels of permission which is associated with different actions. For example, a level 1 authorization would mean people could access information on the servers. Level 2 would allow people to alter data sets and programs. Level 3 would be allowing a user to make system changes and/or delete things from the server.  Accounting is the idea of monitoring the system and data usage during the different sessions of access/utilization. We would then tabulate the data to see the different trends across the company and we could then alter people's levels of access. This would also let us see when there was peculiar behavior in the system, which would then allow us to make changes as needed to correct the abnormalities. |
| **9** | Unit testing is testing code for proper functionality and behavior. This concept will be included throughout the DevSecOps life cycle whenever new code is being generated/implemented.  In this example, I took the idea of creating “white/black” lists in regards to user input into a system. While both processes yielded better results than having no checks on user input, only having a “black” list of inputs that are not allowed still permitted user access to systems or created “surprise features” in the system.  Utilizing a “white” list of only acceptable inputs means that the system will default-deny any input other than what is on the pre-approved list. This decreases the likelihood of “surprise features” exponentially more than the usage of a “black” list. |
| **10** | Automation of our DevSecOps plan will be used for enforcement and compliance with the predefined standards that have been laid out in the DevSecOps attached document.  We will be utilizing some of, if not all, the tools that were talked about in the attached document to help bolster our security efforts. These tools will be used during the pre-production and production cycles coupled with unit testing for the roll-out of new code. During the maintenance phase of the life cycle, we will continue to use the tools at our disposal as well as monitoring exception and error logs to determine where improvements are still needed. |
| **11** | There are two ways we can try to implement security changes to our code. We can do it now or later.  If we act now, we are able to mitigate different exploitations and vulnerabilities from seeing full production and, even better, not let them be seen by the end user. This also gives us a chance at early detection of unforeseen vulnerabilities through unit testing and resolution of initial weak points. Finally, it is easier to do the job right the first time than having to go back through tens of thousands of lines of code to find a single error.  If we “get to it later”, then we are able to pump out code/software “faster” if we do not care about the security portion of the program. We run the risk of leaving exploits open for hackers to steal information and damage the system. Information and data can be leaked before the source of the issue is found, and then we are too late at that point.  I recommend taking security seriously and acting immediately to overall expedite the generation of quality code and keeps the end-user's information safe. |
| **12** | My recommendations for the DevSecOps program overhaul would be to implement a 2FA system with designated privilege levels for each user account which are audited periodically to ensure that all privileges are up to date. Keeping an edit/upload log with timestamps that shows us where and when information has been accessed by the team to prevent internal attacks. Utilizing periodic unit testing throughout the development life cycle gives as many opportunities to catch exploits/vulnerabilities as possible, which also tests the code itself for overall functionality. Once the program goes live, then periodic checks to monitor error and exception logs to track when/if updates need to be performed. |
| **13** | In conclusion, incorporating the different tools and steps previously mentioned help to secure current and future projects from vulnerabilities and increase the accountability we can hold our personnel to. Layering these different practices and tools together will help to round out the company’s DiD strategy moving forward; which, in turn, will help expedite the development and maintenance of all current and future projects. Thank you for your time. |
| **14** | This is the available reference from which the previous standards and policies were adopted from. This is a great website that gives insights into the commonality of different vulnerabilities and how best to mitigate them. |