# CS 405 Project Two Script Template

Jacqueline Woods

[**https://www.youtube.com/watch?v=icQgWOqa0vs**](https://www.youtube.com/watch?v=icQgWOqa0vs)

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

| **Slide Number** | **Narrative** |
| --- | --- |
| **1** | **Welcome to the Security Policy Presentation!**  **Brought to you by Green Pace and Developer, *Jacqueline Woods*** |
| **2** | **DevOps security policy was needed to protect Dev Ops from external and internal technological threats as well as to provide an outline for developers to follow to create secure, functional code! Following a multi-layered approach unit-testing, validating user input, and implementing proper coding techniques will be used to create a multi-faceted security program. The Image on this slide further explains how Dev Ops plans to implement its take on a Defense-in-Depth security protocol.** |
| **3** | **This next slide shows a Threat Matrix featuring the many security threats that are facing Dev Ops varying in severity and likeliness. Most of these threats are completely preventable and all these threats can be blocked within the development phase! Some of the higher risk threats facing Dev Ops include SQL Injection Attacks and Malicious User Input. The other threats featured in the matrix are not as severe but that does not diminish their importance. Improper coding techniques as well as failure to improper memory protection can result in costly consequences.** |
| **4** | **This slide displays Dev Ops secure coding standards and how they relate to our secure coding practices. Please take a minute to review the table and see how each practice is connected to one or more secure coding standards. If a developer were to follow each of these standards and practices, they will have adopted a secure programming protocol as well as a set of effective quality assurance techniques!** |
| **5** | **Here we are looking at Dev Ops encryption policies regarding in-use encryption, in-flight encryption, and in-place encryption. In flight encryption is the encryption of data that is being sent to another network or device. This type of encryption also follows one of Dev Ops coding practices to “Sanitize All Outgoing Data”. In-use encryption is like in-flight encryption as the data that is being encrypted is active. However, the main difference between these two types of encryption is that in-use encryption is meant for data that is being used by a system or user. Cases of in-use encryption is password verification and the processing of financial information. Lastly, in-place encryption is for data that is being stored in memory and is not active. This is for stored sensitive data like employee information.** |
| **6** | **This next slide exhibits a chart about how user access should be handled following the three A’s: Authentication, Authorization, and Accounting.**  **Authentication is the act of verifying and validating all user and employee input data. All usernames, passwords, emails, and security questions must be authenticated before granting access to a Dev Ops public or private program or system. In the case of remote use, the network that is requesting access must be fully authenticated before being granted access.**  **The next “A”, is Authorization. A user must provide authorization before having access to any private or public Dev Ops system or program. Internal authorization can only be granted by upper management and Human Resources depending on the level of authorization. We follow the principle of least privilege here at Dev Ops, which ensures that no one is granted access for more than what is required to do their necessary tasks.**  **Lastly, accounting is an important protocol that is also strictly enforced at Dev Ops. Accounting ensures that the right people are granted access while the wrong people are not: each user and employee account is accounted for and have the correct level of access.** |
| **7, 8, 9** | **The next topic of this slide show is Unit testing. Yes, the dreaded Unit Testing. Dev Ops suggests that while implementing a policy of unit testing it is important to also require a line coverage of 98 to 100%. This ensures that all of the necessary lines within a program have been fully tested, which will limit unpredictable behavior and deeply hidden bugs.**  **This program depicted in the next slide is a simple Task Service program which is centered around user input. So, allowing improper user input can have disastrous consequences for the program as well as for the company who owns the program. The user input for this program was limited regarding its length as well as its contents. Unit Tests were then created to test the programs vulnerabilities with Line Coverage of 100%. Assert statements were used to ensure the correct errors were thrown when improper input was entered into the program.**  **The next slide (slide 8) within the Unit Testing Topic shows how assert throws can be used to test how a program handles illegal input.**  **In the last slide (slide 9) of the Unit Test topic, depicts how the illegal input was caught and handled within the program to prevent its negative effects. In this program an error log is kept to track the errors that are thrown and where.** |
| **10** | **This slide depicts the automation protocol that Dev Ops recommends. In this protocol, automation and analysis tools are implemented within each stage of a program’s life cycle. The next slide will further explain some of the automation tools that Dev Ops recommends.** |
| **11** | **Some of the tools that Dev Ops recommends include: CPP Check, Jet Brain’s IntelliJ IDE automated debugging tools, Unit Tests, built-in tools within Visual Studios, PVS Studios, SQL Inject Me.**  **Dev Ops also stresses the importance of following compiler warnings, and code that disables compiler warnings will not be tolerated unless the developer has received written permission.**  **SQL injection tools will be used in the building and testing phases as well as during the Monitor and Health Check Phases.**  **Creating secure and sound code is of the utmost importance here at Dev Ops and developing code that “cuts-corners” or is insecure.** |
| **12** | **In this slide is a type of risk/reward chart. The major threats that this company is facing includes SQL injection attacks, malicious user access, and improper coding practices. The only major risk in investing in the protection from these threats is the money and resources that would be used to implement them. A minor risk to implementing protection against these threats would be user-end consequences like training employees how to use the new system, and the minor problems that occur any time a new system is released. On the other hand, the reward for protecting the company against these attacks is much greater than the risk. A major benefit would be a peace of mind that your company is protected against malicious users and sloppy coding techniques. Also, the money saved from not facing the consequences of one of these threats is much larger than what it will cost to protect the company against them.** |
| **13** | **It is my recommendation for this company to implement a companywide best coding practice, that is accompanied with employee training on these best practices.**  **A Defense in Depth Security program that uses elements of the coding practices along with limiting user access and data encryption.**  **Finally, to implement Automated Coding Tools to aide Developers write secure and efficient code.** |
| **14** | **It is also my recommendation to have developers create programs with an emphasis on data security and predictable programming.**  **To train developers and employees on how to use automated tools properly adopt a policy where all programs must be peer reviewed before they are submitted.** |
| **15** | **Some of my references include the CERT C++ Coding Guidelines, The Zen Python Best Practices, and the article a “Practical Example of In-Use Encryption”. Thank you for taking the time to listen to my presentation today.** |