# CS 405 Project Two Script Template

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

| **Slide Number** | **Narrative** |
| --- | --- |
| **1** | Hi, and welcome to a security policy presentation. I’m Danielle O’bier and this presentation will be looking at a specific case study and how it relates to security. |
| **2** | Defence in depth is a strategy of building multiple layers of defense to stop or delay attackers.  The security policy I will introduce today is the friend finder network incorporated. This website ran a bunch of other websites that were hacked. It exposed names, emails, and passwords of users over a 20-year span.  If they had successfully used defense in depth in building these websites than the attackers would not have been given the opportunity to collect over 20 years of information. |
| **3** | Most users used their military or government email to make accounts. The worst threats involved the safety of the consumers as well as their credit information. Other threats included these individuals being blackmailed into government espionage, the websites were breached for data, threat factors included intimidation, unwanted attention destroying careers, all the way to potential viruses. |
| **4** | Validate input data, suspicious data should be checked. Heed Compiler Warnings, pay attention to warnings from the compiler.#3. Check that your design is visually and securely safe. Keep it simple, Simple code and simple design. #5 refers to permission standards. 6. Least privilege refers to access levels being only to the smallest amount of necessary users. 7 ensures all data is secure. 8 practice defense in depth is the use of multiple layers to ensure less chance of hacking. Effective quality techniques include fuzz testing, source code audits, and penetration testing.#10 refers to following a list of security requirements so nothing is forgotten. |
| **5** | I broke the coding standards from highest priority to lowest. The ones with highest priority are writing valid strings, understanding data model, distinguishing between characters and preventing SQL injection. Medium risk include don’t define an already named identifier, finding errors and striving for logical completeness. The lowest priority include honor exception, not leaking memory and adding diagnostic tests using assertions. |
| **6** | Encryption in flight- This is data that is stored and protected requiring an encryption key to decode it that only authorized personnel possess.  Encryption at rest- This is data that is moving through the network but can still be accessed.  Encryption in use- This is data that is used on a daily basis. It is usually stored on a database that is accessed thru apps or programs. |
| **7** | Triple-A policies are:  Authentication- confirms the identity of the username and password  Authorization- gives permission to access certain parts based on read and write access  Accounting- refers to billing for service |
| **8** | The best code vulnerability to choose between expect and assert is determined by its use.  Assert when failure should result in termination.  Expect when failure should notify. |
| **9** | The above image is a DevSecOps pipeline. Starting at the pre-production side you read them in counter-clockwise fashion. First Assess and plan, design, build and verify and test. Once that is complete you move on to production and follow in a clockwise fashion starting at transition and health check, monitor and detect, respond, then finally maintain and stabilize. |
| **10** | The DevSecOps pipeline in the previous slide showcases the best way to securely produce code. Following the steps in the model along with using tools that find additional errors will result in a successful product. The two tools mentioned are CPPcheck and Clang static analyzer one being used for PC and the other for MAC. |
| **11** | Some risks include intimidation, blackmail, extortiong for money.  The threats include data breaches, security breaches, viruses.  If they had caught the leak early on it could have stopped many leaks and given them more chances to add to their defense. |
| **12** | Preventable security policies include validating the input data, architect and design for security policies, and sanitizing data sent to other systems.  The hacker could have possibly gained access using a customers account. The authentication should be looked at again for code that leaves easy spots to gain access. A Buffer overflow may have gone undetected allowing an attacker to gain access. |
| **13** | The defense in depth should have been considered. If they had built multiple layers the hackers would not have been able to get as much information and allowed the developers to see the hack sooner. |
| **14** | If you would like to view more resources on the case please visit the websites mentioned here. |