# Fernando Lomeli

12/15/24

# CS 405 Project Two Script Template

https://youtu.be/\_ngYiWfdL9o

| **Slide Number** | **Narrative** |
| --- | --- |
| **1** | Good afternoon. My name is Fernando Lomeli. Today we will be going over Green Pace’s security policy. |
| **2** | Green Pace's security policy showcases a set of principles and coding standards that align with their security objectives and reflect industry standards and best practices. It is needed for securing the organization's safety and defending against vulnerabilities. The policy itself is one of the first steps for defense-in-depth as it outlines some of the layers of security. The image below displays an image of defense-in-depth and the possible security layers within. |
| **3** | The threats matrix table displays the likelihood and priority levels of the security risks. The likely section means it is likely or probable that ignoring these standards could lead to an exploitable vulnerability. The unlikely section means that it is unlikely that ignoring these standards could lead to an exploitable vulnerability. The priority section means that these standards are likely to happen with high severity. The low priority section means that these standards are unlikely to happen with low or medium severity. Automation tools like Cppcheck or clang static analyzer can be used to scan through the source code to find violations of these standards. |
| **4** | This section displays the ten principles of this security policy on the left column and the standards that are linked to them on the right. Some of the standards are linked to multiple principles. |
| **5** | The ten coding standards are displayed with the label on the left and the name of the standard on the right. These standards are in order from high to low priority starting from the top to the bottom. |
| **6** | On to encryption policies. We will be looking at encryption at rest, encryption in flight, and encryption in use. Encryption at rest works on data at rest, data that isn’t actively traveling, which could be databases or file servers. This data is important and is very valuable to hackers. The data at rest should be encrypted with complex algorithms such as AES-256. This will make it so that even if attackers obtain the data, it will be nearly impossible to decipher without the decryption key.  Encryption in flight refers to data traveling from one place to another. This data is at risk of being intercepted and stolen. Encryption in flight involves sensitive data being encrypted as it travels to its destination and being decrypted by the authorized receiver. This allows the data to be inaccessible when intercepted without the decryption key.  Encryption in use refers to data being accessed or processed by users or software. Data becomes vulnerable at this stage. Encryption in use addresses this vulnerability by encrypting and decrypting data in real-time. This makes it so that data is encrypted during its entire lifecycle along with at rest and in flight so that no attackers can obtain plaintext data. |
| **7** | Triple-A policies, the triple A’s stand for authentication, authorization, and accounting. Authentication is the process of verifying who someone is or is claiming to be. A user may identify themselves by using proper login credentials such as username, email, and password. New users will be added to the database along with their login credentials. This policy applies because whenever any user logs into the system they must be authenticated and checked with credentials in the database to defend against certain threats.  Authorization is the process of determining what services a user is allowed to access. Once the user is authenticated, they are authorized with certain privileges depending on their status. This, along with the principle of least privilege, should keep the system protected by keeping users out of areas they should not be allowed to access.  Accounting is the process of keeping track of what resources were accessed by who and at what time. Any files accessed by users or changes to the database can be tracked by session time, date, and which user accessed this data. This policy is beneficial when administrators perform audits to check on any unwanted actions. Tracking changes made to the system is important especially when any issues arise. |
| **8** | The next few slides will showcase testing using the Google unit testing framework. These tests will be able to identify different coding vulnerabilities for positive and negative results. The google unit tests were executed in visual studio. |
| **9** | In this unit test example, we verify that we are able to add a single value to an empty collection. First, we test that the vector is empty and then we test that adding a single entry increases the size to one. |
| **10** | For this test, we verify that resizing to a lower number decreases the collection size. |
| **11** | For this test, we verify that using the clear() method erases the collection. |
| **12** | In this example we created a test to verify the out\_of\_range exception is thrown when calling the at() method with an index out of bounds. This is an example of a negative test. |
| **13** | This image displays the DevSecOps pipeline. There are four stages in the pre-production section including assess and plan, design, build, and verify and test. There are also four stages in the production section including transitions and health check, monitor and detect, respond, and maintain and stabilize. |
| **14** | There are some automation security tools that can be used in different stages of the DevSecOps pipeline. For example, in the build stage, automation is implemented in the form of static application security testing tools like Cppcheck to scan code for any vulnerabilities. In the verify and test stage, dynamic application security testing tools can be used to find vulnerabilities through simulated attacks. In the transition and health check stage runtime verification tools can be used to determine whether the system runs as expected. |
| **15** | Some potential problems that could arise are data breaches or DoS attacks. The solution is to take a DevSecOps approach and implement security by taking a shift left approach and implementing security in all stages of the software development lifecycle. Automation can be used for security testing, to find vulnerabilities, and to run compliance checks. Security should be thought of from the very beginning. It is important to act early to prevent threats and mitigate vulnerabilities. The risks of waiting could result in damage from attacks and issues being found late which could be more costly to fix. |
| **16** | All the principles and coding standards should be reviewed and understood by the entire team. The entire team should also understand that they all bear some responsibility towards security. All employees should be trained in security processes and secure coding practices. Ethical white-hat hackers may also be considered helpful as they can find vulnerabilities and improve security against other attackers. For example, a white-hat hacker named Charlie Miller was able to find a vulnerability that allowed him to infiltrate and take control of a Jeep Cherokees’ systems including the brake system. This posed a threat for the company and as a result they recalled 1.4 million vehicles to fix the issue. |
| **17** | To prevent future problems, the shift left approach should be adopted to start including security as early as possible. Automation tools should also be included throughout the SDLC for security testing, vulnerability scanning, and compliance checks. Technology is always advancing, as such security policies should be constantly reviewed and updated. |
| **18** | References |