# CS 405 Project Two Script

https://youtu.be/USi8lC68zEE

| **Slide Number** | **Narrative** |
| --- | --- |
| **1** | Thank you for coming to my security policy presentation. |
| **2** | We’ve all seen it on the news: an organization falls victim to hackers and the personal information of millions of customers is compromised or their data becomes subject to a ransom. While we can’t prevent every attack, having a strong security policy in place can certainly help prevent them. Security policies help guide developers during the construction of a system and ensure that security is built-in from the start.  The concept of ‘defense-in-depth’ refers to having multiple layers of security. The idea is that each layer will help cover the shortcomings of other layers to provide a more comprehensive security solution. Having security policies in place that address each layer, such as those shown in this chart, can ensure that an organization thoroughly guards their systems. |
| **3** | The threats that the coding standards address can be thought of as a function of priority and likelihood. Some threats have a high risk and are likely to occur, making them the first ones that should be remediated. Some threats are unlikely to occur, but the high risk they pose makes them a priority. Other threats are likely to happen, but do not pose as much of a risk and are thus a low priority. There are also threats with a low risk and low likelihood, making them some of the last to be addressed and are mostly a matter of good practice rather than a big security concern.  Automation can be useful to identify issues like these. Static analysis tools can catch weaknesses when still in development or be implemented as part of a CI/CD pipeline. |
| **4** | Many of the secure coding principles correlate to one or more coding standards. While some principles, like adopting a secure coding standard, are broad and apply to many coding standards, other principles like denying by default and least privilege are self-explanatory and stand on their own. |
| **5** | This is a list of coding standards, ranked from highest priority to lowest. This ranking is based off the threats matrix from earlier. The highest priority standards are ones that are both likely to happen and have a high impact. Next are standards with a high impact but are less likely to happen. After than are threats that are likely to occur but have a small impact. Last are threats that are unlikely to occur and have a small impact. |
| **6** | Data encryption policies can be categorized into three groups: data at rest, data in flight, and data in use.  Data at rest is data not currently in use. For example, customer information stored on a hard drive of a server. Any sensitive data should be secured using a NIST approved encryption algorithm, like SHA256  Data in flight is data in transit on a network. This data needs to be encrypted so that it cannot be used if intercepted. When being sent over the web, this can be secured by using the secure sockets layer, SSL, combined with transport layer security, TLS. Internal network traffic can be encrypted at the application level.  Data in use refers to data the user is actively using. Typically, this information would be sitting in memory, unencrypted, while the user is utilizing it. Using secure memory locations or homomorphic encryption, which allows operations to be performed on encrypted data, will ensure that data is secure through its entire lifecycle. |
| **7** | When looking to secure a system, the triple-A framework is a great place to start.  The first ‘A’ is authentication, which refers to controlling who can access a system. This can take the form of a login screen. Additional measures can be added, like two factor authentication, to further validate the user truly is who they say they are. Every secure system should utilize authentication at some level so that malicious actors cannot access the system.  The second ‘A’ is authorization, which refers to controlling what resources a user can access. By following the principle of least access, the user should only be given enough permissions to carry out their function. By default, resources should be denied unless explicitly given permission. By using roles, this process can be made easier by having groups of permissions for a certain type of user.  The last ‘A’ is accounting, which refers to tracking activity in a system. By monitoring who performed an action on a certain resource, we can detect unauthorized usage. This also makes troubleshooting easier. |
| **8** | The following slides have several tests for verifying the correct behavior of a vector when various operations are performed on it, like resizing, clearing values, reserving capacity, and tests for out of bounds exceptions.  These tests can be setup using the Google Test library for C++. Aside from the tests themselves, the code also contains a test class that the tests will run against and helper classes for testing purposes. |
| **9** | This test checks that adding a single value to an empty vector behaves properly. First the test ensures that the vector is empty, with a size of zero. Then it adds a value and ensures that it is no longer empty, with a size of one. |
| **10** | This test ensures that the maximum size of the vector is always greater than the actual size of the vector. First, it ensures the vector is empty, then checks the max size is greater than the current size of zero. Then, a value is added, and this check is repeated for the current size of one. This repeats for sizes of 5, 10, and 15. |
| **11** | This test makes sure that resizing a vector changes the size correctly. First, five entries are added, and the current size is verified. Then, the vector is resized to one element and the test ensures the new size is truly one. |
| **12** | This is a negative test that ensures the code throws an out\_of\_range exception when trying to access a value that is out of range. Entries are added to the vector, then the code attempts to access the element one past the end of the vector. |
| **13** | This test ensures that using the erase function on the vector, using its begin and end values, erases the entire vector. First, some values are added to the vector, then the erase function is called, using the begin and end values. Then the test ensures the vector is empty and has a size of zero. |
| **14** | The diagram on this slide shows a typical process flow in a DevSecOps framework. While being similar to a DevOps framework, it has some key advantages and distinctions. |
| **15** | The DevOps model combines the development and operations into a single model that allows organizations to create, test, deploy, operate, and improve a system. The DevSecOps model takes this a step further, including security at every level, the big advantage over DevOps. In the initial planning stages, security concerns are considered so that they can addressed through the entire cycle. While designing the systems, developers practice best security practices, and can even practice test-driven design, which involves writing the tests before the code. When the code is being deployed, automated processes can be put into place that perform vulnerability scanning and static analysis, assuring there aren’t any preventable weaknesses in the code. After deployment, automated services can monitor the system and check for things like unauthorized access or suspicious traffic. |
| **16** | While the risks from a cyber attack may seem obvious, they can go beyond just the loss of data. The cost of downtime may not be insignificant, and in the case of ransomware, it may cost money to get those systems back online. For financial industries, the target of the attacks may be money itself. Reputational damage should not be overlooked either. A successful attack may give the impression that the target is not serious about security and customers will be less likely to utilize their services in the future. This ultimately becomes financially damaging as well. If the attack happened because of negligence on the target’s part and causes damage to others, they could be subject to fines and penalties from government agencies.  The benefits of implementing security measures proactively are the opposite of the risks. Things like secure and regular backups could minimize the effects of ransomware. Being known as a company with strong security would only increase consumer trust and by doing their due diligence, it would be unlikely they would be found to be neglectful and sanctioned by a regulating body. |
| **17** | While the policies laid out in this presentation are certainly a good start, there are more gaps to be filled. While the coding standards cover many situations, they are by no means comprehensive. Ideally, we could extend them to include examples that relate to all our security principles. Additionally, many of these policies are focused more on the application level of security. However, there are a number of layers that need to be considered for true defense in depth. This could include training users on best security practices, like preventing social engineering. There could also be policies about data management, like what data is backed up, how frequently, and where that data is stored. We could also create policies about network configuration, like what parts of the system are connected to other parts, firewall configuration, request limiting, etc. Even physical security needs to be considered. The National Institute for Standards and Technology has comprehensive standards for things like physical access controls, fire safety, utility failure, and structural collapse that could be used as a basis for such policies. |
| **18** | To summarize, it is important that we do not treat security as an afterthought. Security should be a consideration throughout the entire development and operation process, and models like DevSecOps can help ensure this. Besides being more difficult to implement after the fact, it may be less effective. You can have a strong steel gate, but if your castle is made of sand, it won’t do any good. But in addition to a gate, we need other measures to truly secure our castle. A gate may fail, but intruders may never reach it if we have a moat, a team of archers, and defensive palisades. That’s where the concept of defense in depth comes into play. One security measure may fail but having enough layers in place will help ensure there is no critical failure of the whole system.  By having security policies in place, everyone developing and operating the systems can be on the same page and have guidance on how to implement security. But just as technology constantly changes, so do the treats it faces. Therefore, it is important to constantly reevaluate our polices to make sure they are still adequate. |