# CS 405 Project Two Presentation Script

https://youtu.be/O78SF182DMU

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

| **Slide Number** | **Narrative** |
| --- | --- |
| **1** | Hello, my name is Enrique Zarate and I’m the developer charged with assessing our current security policy and outlining a proposed security policy based on the assessment. |
| **2** | First off, let me talk about Defense in Depth. Cybersecurity and security in general have multiple layers since there are many ways to secure data or assets. We have the physical layer on top that would be like access to the databases and making sure they are in secure locations where outsides cannot get to them. Further in toward the core is the application security which will be our focus today with the discussion of removing as many vulnerabilities as possible from the code. You can notice in this diagram that vulnerabilities can be in many parts of these layers ranging from network security, host security, endpoint security, and app security. Our job is to make sure that each of these layers is as secure as they can be so there are more barriers that will stop hackers from obtaining Green Pace’s or our clients critical assets or system’s data. |
| **3** | There are many guides out there on what secure coding is, so much so that the 10 principles of coding were created. These can be seen here and may be straight forward but when we begin coding and become focused on building a functional product, we can easily forget about these basic principles. Coding standards were pulled from the coding principles and to show how much these go hand-in hand, I’ve listed the coding standards on the right and how many principles they can be a part of. For example, SQL injection can be seen under the valid input data principle, the Architect and Design for Security Policies Principle, the Sanitize Data Sent to Other Systems Principle, and the Practice Defense in Depth Principle. |
| **4** | The coding standards we have put together for us to start with, were then placed into a matrix of higher priority items to lower priority items. The SEI Cert Coding standards were used for this as they have data on which items may be more likely vs not and also the cost associated with these kind of vulnerabilities. Based on that data, we were able to order them as seen in the next slide. |
| **5** | Our Coding standards are as follows with 1 being the highest priority and 10 being the lowest. [Read them one at a time.] |
| **6** | The coding standards themselves are very important in reducing the number of vulnerabilities that hackers can take advantage of but we also need to discuss the need for encryption. Encryption of our data so that outsides cannot access it or read it is another layer we would want to add. There are three different kinds of encryption we need to consider and that is encryption at rest, encryption at flight, and encryption in use. Encryption at rest is pretty general and is needed to keep our database information secure so it is not readily accessible to hackers. Worst case, they are able to steal the information, but they don’t have access to understand it with it being encrypted. Personal Health Information if it is needed by clients will most likely need an extra layer of encryption since these can be a bigger target for hackers. Encryption at flight is needed so all of our communications between each other and between us and third parties are kept confidential. Using SSL or TLS encryption is great so we can avoid man in the middle attacks. Keys or certificates should be handed out only to recognized users so that communications stay between those parties. Encryption in use is the basic idea of only decrypting items that are needed rather than decrypting the whole database. This will decrease the chances that a hacker will have access to the whole system and the only information that is vulnerable is that in use. |
| **7** | The triple A framework is also something that we need to ensure we are implementing to it’s full capacity as another layer. We will be discussing Authentication, Authorization, and Accounting here. Authentication is for making sure the person getting into the system is who they say they are. For example, user name and passwords are given to those who should have access. To make things more secure, other forms of authentication are good to use such as sending a code via text message to that user so only that user is can be verified with a secondary method. Authorization is next and that is making sure users can access only what they need. By default, all users are denied everything. Based on their positions, access to certain parts of the software will be given based on what they need to do. Lastly, accounting is a way to monitor the system. Changes being made or logins all can be checked, dated, and verified. This will help the security team monitor for any weird behavior that may help them detect criminal activities. |
| **8** | Now that we have decided what kinds of vulnerabilities we are worried about, we want to ensure that we are testing for these vulnerabilities and understanding them and how they work. |
| **9** | The first unit test we have is if our compiler will recognize the accessing of uninitialized memory. In this example, We started with a vector of 7 items. We verify that the vector is definitely not empty and then we check to see if there is an error thrown because of the accessing of the item in the 99th spot which doesn’t exist. The error was thrown and we know that we can rely on the compiler to realize this. To make this more simple though, we may just ensure that the accessing of that uninitialized memory is prevented. |
| **10** | In this next example, We try adding some symbols to the argument where a number is normally given on it’s own. It seems like this would be obvious not to do but we as cybersecurity are taught that you can’t really trust what the user inputs so best to try and find out what breaks the system. To take this a bit further, we could attempt to use more symbols in front of numbers. Ideally to avoid this though, we can validate user input here. |
| **11** | Data storage based on data type is important so this test is checking in what cases the data storage may be changed if at all. Since adding data to vectors in some cases can increase the capacity automatically, we may want to make sure that the capacity may not be changed when it’s not full. In this case, we set the capacity at 30, then added 10 items to the vector and then 5 to the vector. The capacity was the same as before showing it had not been changed. An additional test that may have been able to be done here though is checking to see if the spots that don’t have a value allocated led to a vulnerability in which case I believe it may be since it may be something like a null character. |
| **12** | Lastly, we can ensure that the memory is protected by ensuring that all the memory was properly cleared. In this case, we added some items and then cleared it and checked the size of the item. We could have added another step here and made sure that the capacity was also reduced again once the clear occurred to ensure there is no memory that was left open. |
| **13** | Here is the DevSecOps diagram. Currently we have a DevOps process in place with planning, designing and building our programs, testing them and then placing them in production where we are able to consistently check on it. While in production, we can maintain and continue to remove bugs as items are updated add new features. |
| **14** | We already have the compiler that helps us detects weird or funny code that can lead to vulnerabilities, now we can look at other tools that will also help us find things that the compiler cannot. Some of these tools are going to be Parasoft C/C++ Test, Polyspace Bug Finder, Code Sonar, and Astree. These tools were the most mentioned based on the coding standards and when crossed referenced with SEI Cert Coding Standards. |
| **13** | BACK  These tools will best be implemented in the testing phase and the maintaining phase when items are consistently being updated or changed based on changes of the dependencies or updates to other tools being utilized that may open up more vulnerabilities. Although adding these test with these tools can seem tedious, they will help identify leaks in our system that may have larger consequences. |
| **15** | Security is best when it’s included in the process rather than an afterthought. Some of the benefits and risk are listed below. Integrating security into the design early will identify vulnerabilities right away that will reduce the chances for breaches. Focusing on security will help detect attacks before they can be very severe and starting early can also make it easier to maintain a secure program. Waiting too long can be costly in repairs and reputation with the public. Hackers will always find a way into the system so we need to be prepared either way because it’s somewhat inevitable. If we put off security, we are allowing the amount of work to put something in place to pile up. While that is happening, we are also allowing our work and the work of our clients to be unprotected. We need to implement security systems straight away. |
| **16** | There are some gaps that we still have to work out such as finer details of the encryption and the Triple A methods. I have included them in this presentation as something we need to implement but more plans should be put in place for those specific items. With a wide range of talent and backgrounds for our developers, it may be important to re-visit training for coding standards such as the SEI Cert Coding Standards. We may need to discuss the need for other training such as phishing attacks, man-in the middle attacks, and denial-of-service attacks and ways that we can prevent allowing hackers to use those against us or our clients. |
| **17** | I would suggest that our company adopt the Cert Secure Coding Standards altogether as we really should be following them, however I know that much of this is still pretty new to our organization and we need to start integrating it without overwhelming our teams with the many changes that are to come. That being said, I would encourage all of our teams to become familiar with these first 8 that have the highest priority and become familiar with the Coding Standards document being sent out for Green Pace as it will continue to grow over time. I’ll finish with this story about Jonathan James. He hacked NASA out of curiosity of how space stations work. Although Jonathan’s intentions were not bad, they were still illegal and once it was removed from NASA’s networks, who know what kind of security Jonathan may have had on the outside which could have led to NASA’s intellectual property becoming more vulnerable. Had NASA had more security in place, they would have never had a breach and Jonathan may have never been convicted, both parties were at a loss in this one. Security can be helpful to both parties as in this case. |