# Project Two: Security Policy Presentation

Shawn Edouard

2/21/2022

<https://youtu.be/d3B1093pIBQ>

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

| **Slide Number** | **Narrative** |
| --- | --- |
| **1** | Hello, my name is Shawn Edouard and today I’ll be presenting my security policy for the Green Pace project. |
| **2** | So as an overview, this security policy will cover a vast majority of attack vectors that real-world bad actors would use, and will also help ensure your code’s security. With defense in depth, there are multiple layers which you should have when it comes to your own security, and this policy will go over different things you can do to counter different measures. With each layers comes an additional obstacle that an attacker would have to overcome to penetrate your defense |
| **3** | For the threats matrix under the likely category, we have string correctness. This ensures we have sufficient space when accepting input from the client to go into a char array for example. For priority we have SQL injection, and for this we’d want to sanitize all SQL input that the user would provide. For the low priority, we have data value, which will ensure the user gives an appropriate value to the appropriate datatype. This is uncommon since most people, they know whenever they are coding something that they will properly give the right data type rather than the wrong one. For the unlikely one, comes data type which will ensure you are passing matching datatypes to the function. Again, this is highly unlikely to come across but is something you should always keep in mind. |
| **4** | For our 10 standards, we have data type, data value, string correctness, SQL injection, memory protection, assertions, exceptions, variable naming, proper naming convention, and wasteful variables. Alongside them you can see the principles that match up to each of these standards. Overall, it should be self-explanatory that with these defined principles it will help secure your code and improve for the better. |
| **5** | As far as the coding standards, in order from greatest to least priority we have string correctness, SQL injection, memory protection, data type, exceptions, assertions, variable naming, proper naming convention, wasteful variables, and data value. The ones with higher priotiy have a more likely chance of occurring, and are more likely to be able to run arbitrary code, and are usually inexpensive to repair. The ones with a lower priority are typically expensive to repair and will likely never happen. For the highest priority ones, of course since they have a higher priority you’re going to prioritize them and get them fixed as soon as possible since if an attacker were to notice this vulnerability, they could compromise your entire system. |
| **6** | For encryption policies, the policies I’ve submitted do not touch on encryption, but there are three different states of encryption. At rest, the data is stored on your computer and you have to encrypt it. While in flight, it is being transmitted over the internet/network, and as it is being transmitted you are encrypting it. For in use, you are only showing data to an authorized user,. That’s pretty much it. Unfortunately this policy does not touch it, considering encryption is a standard policy that everyone should automatically follow. |
| **7** | For triple-a policies, there area three different types. Authentication, authorization, and accounting. For the policies I’ve submitted, they do not touch on these, however you should still keep these into account. Authentication will properly identify a user, such as username and password. Authorization will ensure the user is authorized to access certain data. For example, a visitor or regular member on a forum site should not be able to ban someone. For accounting, you will log all actions done by a user. So, if an account views a document or decides to ban another user, you will log everything so you can always trace it back to the person who did it. This will help in the aftermath of an attack, or in the detection of an attack so you can setup systems where you can detect suspicious activity. |
| **8** | **Skipped section, honestly forgot about it.** |
| **9** | For automation, this is a diagram depicting the pre-production and production phase. What we will discuss on the next slide is how you should implement these polices and at what phase you should do it. |
| **10** | Security should be kept in mind during all phases, whether you are planning, writing the code, or after it is already in production. The reason is, you always want to keep in mind certain protocols, such as never to trust the client. If you implement this too late, then you will have to go back and redo all the hard work to ensure it all follows the protocol. As far as automation, it would likely take place at the end of the pre-production phase since you’d want to run the appropriate checks after your code is made, but before your code is in production. This will ensure you are following your policy and will have no issues in production. |
| **11** | For the benefits, the benefits of acting now is that you will have aa secure system as soon as possible that should be immune to a majority of attacks that bad actors could act on. You will quickly plug up and patch all vulnerabilities which will give you a safe system and a safe mind. The risk of deciding to wait is that you are leaving yourself proine to basic cybersecurity attacks such as brute forcing a password, SQL injection, and more. Why would you wait to secure a system that holds confidential information. |
| **12** | The main gap in this security policy includes encryption and having common sense. As far as common sense, one has to realize that common sense is not so common. It would be a great protocol or standard to have where you enforce strict password requirements and require at a minimum a monthly password reset for all accounts that have substantial access to where if an attacker acquired it, they could do great damage. For the Colonial Pipeline ransomware attack, the intruders gained access through an unused account with great access. While it is not 100% confirmed on how they got the password, it is the general consensus that it was a compromised password on one of those password leaked list, which meant that the account owner shared the password on other services and did not change it frequently. This vulnerability lead to an easy gateway into the pipeline and could’ve lead to so much more damage caused by the attackers. As far as the encryption, this is a standard industry practice that everyone needs to be followed. I personally feel this shouldn’t need to be addressed in a security policy, but common sense is not so common. This is a future standard that someone should implement. To prepare for the future, this policy should always be open to policy edits and additions. The latest widely-recognized security policies (ex: NIST Cybersecurity Framework) should always be considered as well as constantly educating your employees on the best security practices. |
| **13** | Overall, the standards that need to be adopted are from the third standard down to the 45th. These have the highest priorities, and they need to be adopted as they all address a great security or programming concern. |
| **14** | Here are my references below for the colonial pipeline and the SEI CERTC++ coding standard that I used. Have a good day. |