# CS 405 Project Security Policy Presentation

Nicholas Newlin

10/18/2021

https://youtu.be/4qcfEdk-Dxo

| **Slide Number** | **Narrative** |
| --- | --- |
| **1** | Our security policy is vital to the security of our users and systems. We utilize Defense in Depth, which means we have multiple layers of defense. The security policy will support this defense in depth by outlining secure coding practices, defense strategies, testing processes, and many other security measures we will implement. Each layer of defense helps ensure the system is more secure. |
| **2** | The first, and likely type of attack is unauthorized access to our system. As the primary line of defense into the system is user authentication, it is important to verify users are who they saw they are. Unfortunately, many people do not follow proper password protocols and use either easy to guess passwords or the same passwords for multiple systems. This makes it important to be able to add another layer of security to the login process in order to prevent unauthorized access.  Another threat, which is a priority, is SQL injection. The system is based on a SQL server, which means it is vital importance to ensure proper sanitizing of requests to the server to prevent SQL injection.  Another threat, in the low priority category, is Buffer Overflow attacks. This threat is categorized as low priority as we utilize unit tests and automation to detect buffer vulnerabilities during the coding process. This will be discussed more further on in the presentation.  The last threat is categorized as unlikely, and that is Exception exploits. |
| **3** | These are the 10 principles. And how the coding standards align to them. The standards will be discussed in the following slide  [List principles] |
| **4** | Our encryption policies cover all states of data. The first is data in flight. This means data that is being moved from one system to another. During this state it is important to ensure it is protected with encryption in case it is sniffed or intercepted during transmission.  The second state is data at rest. Any data being stored, whether in a database or on a server’s hard disk, it needs to be protected with encryption. This ensures any unauthorized access cannot read the data which is stored or breached.  The last state is data in use. Any data that is being manipulated will only be unencrypted when absolutely necessary to access the data. This unencrypted data will not be stored or sent to another system while in the unencrypted state, but will be first encrypted. |
| **5** | Triple A policies stand for Authentication, Authorization, and Accounting. Authentication means verifying someone is who they say they are. This is done with a system such as a login via username and password. It can also include 2 factor authentication to provide a deeper level of security when verifying they are really the correct person.  Authorization is ensuring they are authorized to view or access what they are requesting. This is not the same as simply logging in, but is granting them the proper access that they require, without giving them more than necessary.  Accounting is keeping records and track of all requests made. This helps identify potential malicious attacks and develop defenses for future attempts. |
| **6** | This is a simple string sanitization function created to show how unit testing works. This function takes an input string for an SQL query and checks for potential injection. |
| **7** | This shows the test for a valid simple SQL query string. This string does not contain any malicious intent, and should return the bool injectionFound as false |
| **8** | This shows another valid string. As this does not contain malicious intent either, it should return as false as well. |
| **9** | This test includes malicious intent for injection by adding the substring “or 1=1;”. This test should set the bool of injectionFound to True. |
| **10** | This test’s string includes malicious string utilizing two strings equaling each other. As with the first malicious example, this should also flag the bool as True. |
| **11** | As you can see from the results, each test case has passed the Unit test. This example, while simple, shows how unit testing can be utilized to test for secure coding practices. |
| **12** | Different automation methods will be used at different stages of the development process. One method of automation will be compiler errors and warnings. This will be used in the build stage. While creating the code, any warnings and errors created within the compiler will be addressed.  Another form of automation is utilizing static testing. This will be used in verify and test phase, as well as utilizing unit testing. This will ensure that any code is safe and error free after the build phase. |
| **13** | When implementing a security policy, it is important to understand the risks and benefits. While it may seem time consuming to implement these systems during the development process, it can greatly reduce time consuming rewrites of code. Another risk, is longer development times. While this is seemingly true, it is necessary to take into account not only the time of start to product release, but potential incidents that must be addressed after release. When this is factored in, the reduction in security issues due to implementing these policies will greatly reduce the frustration and hours spent patching holes as they appear. Another risk in not implementing these policies is loss of secure data which could lead to loss of trust in the company.  On the opposite side, the benefits of implementing these policies now are the releases will be more secure. Spending the time now to implement security means the released product will be inherently secure. Rewrites of code to patch security vulnerabilities will also be lowered. These will all combine to increase confidence in the users. |
| **14** | It is necessary to ensure we utilize modern encryption on our data at all times. In 2018, data including payment methods for millions of users was exposed due to poor encryption practices.  Another recommendation is reducing the chances for human error. All the secure coding in the world do not matter if someone gains access to login and password information improperly. In 2018 someone in Pennsylvania’s Office of Administration improperly assigned permissions to users that allowed users to see personal information for other users. This data breach was not due to a breach in security, but a user making a mistake exposing multitudes of user data. |
| **15** | In conclusion, it is necessary to follow the 10 principles. This also means implementing and following the secure coding standards. These are designed to help identify potential security vulnerabilities in the system before the can be exploited maliciously. Also, defense in depth best practices must be following throughout the entire life of the system. Simply implementing secure coding processes is not enough if other security policies are not followed. All levels of security are vital to ensure the safety of the data.  Next, this means that the data needs to follow encryption standards. All three stages of the data lifecycle must be utilized: in flight, at rest, and in use. Finally, the Triple-A framework needs to be followed. Properly authenticating, authorizing, and accounting are necessary to provide security to the system. Without verifying who and what is accessing the system, any other security measures taken are invalidated. |