Daniel Coffey

6-1 Journal: Don’t Leave Security to the End

CS 405 Secure Coding

02/21/2022

It is best practice not to save writing tests for last. When I can, I try to implement tests before I even code anything. This is known as test-driven development (TDD). TDD focuses on the bigger picture, which leads to more functional software. There are numerous benefits to TDD, and they are it permits for a rapid conception to determine if the code base has any functional issues, it is documented in the codebase of functionality, and it allows for safe refactors of code. (Farinelli 2020)

Utilizing TDD means that the code base will be maintained and cleaned up often. This will provide a proper platform for new test and code to be implemented. This will greatly increase the structure of the software, especially during any new feature implementation or maintenance. At the heart of TDD are unit tests, these tests are small, but they will keep your code inline and make sure that your code is more modularized and flexible.

Writing tests as you go along, such as unit tests, will ensure that nothing has been missed. Waiting to write tests at the end, especially in an application with a lot of files can increase the chances of something being missed. This could lead to an open vulnerability and point of interest during a cyber-attack.

To make sure that your software and information is protected against any potential threats, you need to make sure you understand the different types of threats and utilizing the best secure coding practices.

There are certain rules to follow to prevent cybersecurity threats. One of the rules is to establish software design requirements specifically to prevent and mitigate cybersecurity threats. Another guideline is to enforce secure coding standards for cybersecurity threats. The last guideline is to test early for cybersecurity threats. You should always test your code early and often to identify potential vulnerabilities in your code that can lead to a cybersecurity attack. (Ashley 2021)

In my Project Two presentation, I made sure that security is addressed inherently and not left until an issue is discovered. I did this through various ways. I incorporated defense in depth, which is a series of defense mechanisms that are layered to protect data. Essentially, if one mechanism fails another will immediately take its plays. This creates more security for all different types of attacks. I then incorporated a threat matrix to prioritize potential threats in an order of urgency. I also implemented the 10 coding standards and the 10 principles. Encryption policies were also incorporated into my Project Two presentation. The encryption policies cover encryption in flight, at rest, and in use. I also included the Triple-A policies which are authentication, authorization, and accounting. I then went over the unit test examples that I have done. Went over the automation summary and went over the tools such as the DevSecOps Pipeline, which refers to integrating security into the software development life cycle.

**References**

Ashley, M. (2021, February 12). *How to Prevent Cybersecurity Threats With Secure Code*. DevOps.Com. Retrieved February 20, 2022, from https://devops.com/how-to-prevent-cybersecurity-threats-with-secure-code/

Farinelli, D. (2020, March 2). *The Benefits of Test-driven Development*. DevOps.Com. Retrieved February 20, 2022, from https://devops.com/the-benefits-of-test-driven-development/