Daniel Coffey

8-2 Journal: Portfolio Reflection

CS 405 Secure Coding

02/27/2022

This course has taught me a lot about secure coding. I have completed various assignments ranging from numeric overflow coding, defense in depth, SQL injection coding, buffer overflow coding, exceptions, unit testing, encryption coding, static code analysis, not leaving security until the end, and discovering the motives for the attack. I have learned that numeric overflow coding occurs when an arithmetic operation tries to create a number value that is outside of the range that can be represented with a given number of digits, either higher or lower. Defense in depth is a concept used in security where multiple layers of security controls are places throughout an entire system. The purpose of this is to create a redundancy incase a vulnerability is exploited. The idea behind this is that if one line of defense fails, another will immediately come and take its place. SQL injection is a code injection technique that might destroy your database and it usually occurs when you ask a user for input, such as their username, which gives you an SQL statement that you will unknowingly run on your database. Buffer overflow is a condition that exists when a program attempts to put more data in a buffer than it can hold or when a program attempts to put a data in a memory area past a buffer. An exception is an event that occurs during the execution of a program. The exception will then disrupt the normal flow of the program’s instructions. When an error occurs within a method, the method will then create an object and hand it off to the runtime system. Encryption coding is method used to encrypt a message where the message may only be viewed by utilizing a key to decrypt the message. This will keep the contents on the message secure and a secret to anyone without a key who tries to view the message. Static code analysis is a method of debugging by examining source code before a program is run. This is done by analyzing a set of code against a set, or multiple sets, of coding rules. Lastly you want to understand the motives for a potential cyber-attack, such as financial gain, notoriety, etc.

One of the important things that I learned was to have security from the start. This essentially means that you want to ensure that you are including the security principles and policies from the beginning of the development cycle. There are a few things to consider while planning security from the start. They are to make sure you define the boundaries clearly, trust no one, foster a security culture, understand your baseline, start with segmentation, enable automation, review security and privacy by design principles, and secure your storage systems.

It is important to evaluate and assess the risk and cost benefits of mitigation. Risk plays a huge factor in decisions and choices that are going to be made. A software vulnerability is a weakness in the software that compromises the overall security of the system. This can be a problem as a data breach or system attack can cost millions of dollars in compromised files, operational challenges, and system fixture and maintenance. It is more cost effective both financially, and in time to prevent an attack than to try to repair damage from an attack. There are multiple ways a system can be attacked such as an injection flow, broken authentication, security misconfiguration, and use of components with known vulnerabilities. It is crucial to test early and often. Testing also permits for a rapid conception to determine if the code base has any functional issues; it is also documented in the code base functionally, allowing for safe refactors of code. At the heart of test-driven development 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 unit tests as you go along, will ensure nothing has been missed. If you wait to test to the end, you may miss something and lead to an open vulnerability and point of interest during a cyber-attack. You may also implement penetration testing, which is an attempt to evaluate the security by trying to exploit vulnerabilities. Once you have tests set in place, you may use automation to test regularly and efficiently.

Zero trust describes the need to move security leaders away from a failed perimeter-centric approach and guide them to a model that relies on continuous verification across every device and application. This means going from a trust but verify approach, to a never trust and always verify approach. This iterates the concept that “no one is safe”. I agree with this take because even the most secure platforms can get breached so we need to make sure that we try to stay on top of everything. Ensuring that every part of the system is strong, will create a better ecosystem of security. Zero trust ensures that verification is always going to be utilized. This can be done by various means such as multi-factor authentication to identify a user. Zero trust is effective because it is better to be safe than sorry. It is smart to assume the worst and be prepared for any kind of attack, then not be prepared, and have to fix a successful attack. Ironically, if zero trust isn’t implemented, users might have zero trust in the software/company if security gets breached.

Security policies need to be integrated into all life cycle stages of application development. These stages are the requirements stage, architecture and design stage, implementation stage, testing stage, deployment stage, and maintenance stage. While utilizing secure coding standards, you can prevent any security vulnerabilities. They will prevent, detect, and eliminate the errors that may cause a problem and compromise software security.

**References**

Bellairs, R. B. (2020, February 10). *What Is Static Analysis? Static Code Analysis Overview*. Perforce Software. Retrieved February 27, 2022, from https://www.perforce.com/blog/sca/what-static-analysis

Editors, Forbes Technology Council. (2020, April 10). *11 Ways To Address Security From The Start Of Your Tech Project*. Forbes. Retrieved February 27, 2022, from https://www.forbes.com/sites/forbestechcouncil/2020/04/10/11-ways-to-address-security-from-the-start-of-your-tech-project/?sh=5ca3029569a8

Foster, S. F. (2020, October 22). *Secure Coding Practices: What Are Secure Coding Standards?* Perforce Software. Retrieved February 27, 2022, from https://www.perforce.com/blog/qac/secure-coding-standards#:%7E:text=Secure%20coding%20standards%20are%20rules,that%20could%20compromise%20software%20security.

*How To Mitigate the Risk of Software Vulnerabilities*. (n.d.). Blue Coding. Retrieved February 27, 2022, from https://www.bluecoding.com/post/how-to-mitigate-the-risk-of-software-vulnerabilities

*Secure Coding Practice Guidelines | Information Security Office*. (n.d.). Berkeley Information Security Office. Retrieved February 27, 2022, from https://security.berkeley.edu/secure-coding-practice-guidelines