Dane Clark

04/17/2023

CS-405 Secure Coding

Instructor: Aaron Demory

Journal Reflection

Throughout this course we have learned how simple it is to leave our code vulnerable to attack. This has become an eye-opening experience for me. Using improper data types and values are some things that I know for sure I have been guilty of in past coding projects. Understanding why these are crucial areas of concern has been the thing I will reflect on the most. Adopting the coding standards are great ways to prevent issues before they arise. However, we will miss some issues inevitably. This is why it is important to test our code early and often. It sounds simple but it is surprising how easy it is to forget. The advantage of the strategy is we will mitigate the amount of error, and there for vulnerabilities, that come up later in development.

This is directly tied to the amount of cost it would be to leaving security to the end of development. By leaving security to the end of development we allow a lot more room for error. If we are dealing with millions of lines of code, debugging and testing will cost much more since we must sift through much more code. Additionally, when an issue is found correction needs to take place, thus adding more time to development. Naturally, focus will be put on the more dangerous threats. While this is important, many of the smaller issues will be overlooked as they do not typically stand out as obviously as major security risks. This is where our organization will face the biggest threats. Often time it is something “small” that can lead to security breaches. The cost of fixing issues after an attack has happened is much more than just money. Reputation as well as public trust can be damaged beyond repair. The initial breach could cost millions, and then the aftermath could continue to cost the organization money as they enter damage control.

Even if our coding has followed all the steps necessary to prevent attacks, we still have the issue of outside attackers gaining access to our programs and systems. This is where Zero Trust comes in. Instead of building up was around company data, we instead build walls around what is trying to access this data. Users, devices, and sessions will all be verified before access is granted. This is important as there are many more ways to authenticate users and their devices today. Multi-factor Authentication, and biometric authentication are great ways to ensure that the right access is granted to trusted people, and devices. Implementing a zero-trust strategy is advantageous to organizations and their employees as it provides both with a greater level of security. Additionally, zero trust makes it much more difficult for attackers to gain access to areas of the organization without the proper credentials and authorizations. We can think of zero trust as a series of checkpoints that the user must pass to gain entrance. Making sure every layer of the organization is protected with its own checkpoint adds even more security.

None of these matters if we do not have the proper policies in place for our organization to abide by. Implementing a security policy is how we ensure that our whole team understands what is expected of them from a security standpoint. A security policy should be easy to understand and easy to implement. Some recommendations for how this would work would be to keep it simple. This means that our code should be easy enough to understand. We will not be the only ones viewing this code, so making it easy to understand is an advantage as we move through the development life cycle. It also makes is easier to find mistakes or potential vulnerabilities. Adhering to the coding standards is a given. We much do this to ensure that we are not introducing vulnerabilities into our programs. The coding standards help mitigate many risk factors that arise during the development life cycle. Lastly, check early and often. This cannot be stressed enough, but checking code early and often is the best way to protect our organization from attacks. Using software to check our code through automation is a great way to find issues we might not see right away. There are a lot of things that an IDE would miss so using special software such as CPPCHECK is another great way to ensure that we are following our security policies.

Secure coding is one step in protecting ourselves, our organization, and our customers by ensuring that we are taking all the necessary steps to prevent attacks. Coding standards, zero trust implementation, and developing a security policy all work in tandem with the idea of secure coding. Our priority as developers should be the safety of everyone using our products, but also to ensure the continued growth of our organizations. Security breaches and attacks are ways that will both damage the brand but also may potentially put our users at risk. Adopting a secure coding mindset gives us the best possible defense against these risks.