**Jacqueline Woods**

**May 23, 2021**

**Module Eight: Reflection**

**Reflection of Security Practices and Standards**

**Don’t Leave Security To the End**

The programming best practice “Don’t Leave Security to the End” is a vital practice in ensuring the security of your program. The statement means that a developer should have their program’s security in mind at all stages of development and not just in the end after a program’s vulnerabilities have already been potentially exploited. This last-minute coding practice is reckless and will cost the developer more time and resources than it would have to just implement security practices during each stage of the development process. . Frankly, there is no excuse to leave security to the end of development, especially when it has become so easy to implement security practices during development. Automation tools like CPP check and SQL Inject Me, allow the developer to easily implement secure coding practices and defenses.

**Zero Trust**

Following a Zero Trust policy is one of the best ways to safeguard our program against malicious users. The concept of zero trust will provide an extra layer of protection over already standing good security practices and protocols. The zero-trust concept places a network's security on its user access and not just on the network’s security measures.  It is impossible to make a network completely impenetrable, however by incorporating zero trust, it will make it much more difficult for an unsolicited user to gain access to the network because they will now have to pass a series of user security protocols on top of getting through the network's other security measures. Although, zero trust does make the network more secure it does make accessing the network for legitimate users more difficult. This could frustrate the user, or it could make the user feel more secure, I believe if incorporating a zero-trust security concept, it will be important for it to flow in a way that does not interfere with valid network use. By enabling things like two step verification or answering a security question are ways to accomplish this without interfering with valid user access in a way that is detrimental to the user. It’s a simple extra step that can prevent spending millions fixing a security breach and consequential data leak.

**Prevention vs Mitigation**

Many secure coding experts believe in the idea “Prevention is better than mitigation”. This practice revolves around the idea that it is better to work to prevent security threats than it is to just wait until an active threat exists. It is not only easier for a developer to prevent threats than work to mitigate the damage of an active threat it is also cheaper for organizations to develop their code in this fashion. Basically, a lot less resources will be used to prevent a threat than will be used to mitigate the damage caused by allowing the threat to turn into an attack.

**Adoption Of A Secure Coding Standard**

The adoption of a secure coding standard is vital to a developer’s success. It is crucial that a program’s security is being thought of during each of a program’s development phases. If a program is secure down to each line of its code, it will be much less vulnerable to potential attacks. As discussed in previous sections of this reflection it is much easier and cheaper to develop secure code than it is to mitigate the damage done by vulnerable program. Adopting a secure coding standard will save a developer from the headache of trying to cover up vulnerabilities after a program has already been released. By using a set of secure standards and principles to help guide will make you not only a better programmer but a more valuable one as well.

**Security Recommendations**

Implementing a defense in depth policy is a good place to start when it comes to threat prevention. Some of the steps that can be taken to build a defense in depth policy include employing a strong firewall for network protection, use a VPN for safe internet browsing, use anti-virus software and scan your systems regularly to check for viruses as well as malware, run periodic checks that check a system for any sort of abnormal behaviors, adopt a set of secure coding standards, and use effective automation tools to sniff out vulnerabilities in code. There is no one-size-fits-all approach when it comes to adopting a threat prevention policy but a mixture of any one of these defenses is a great place to start when it comes to threat prevention.