**8-2 Journal**

by Brandon Hollada

Southern New Hampshire University

CS-410-12252-M01

Submitted to Aakhil Gudivada

Throughout this course, I have learned that security can’t be an afterthought. Security must be implemented and a priority throughout the entire development to help limit vulnerabilities and to be able to respond when they are discovered. A large portion of these issues such as buffer overflows, improper input handling, and missing tests can be prevented through secure programming practices(Seacord, 2013). These practices begin during the design and development phases with ensuring the security is being implemented into the design. Doing this early on reduces the risk of costly changes that would be required if vulnerabilities are discovered closer to deployment or afterwords.

Evaluating and assessing risk is another important component of effectively securing coding. Not all vulnerabilities pose the same level of risk and resources(time and money) require the developers to determine the importance of addressing the vulnerability. Developers need to consider the likelihood and impact of a vulnerability being abused to determine which to address first and before deployment(Seacord, 2013). This mindset helps provide a secure code, while allowing organizations to maximize resources. I have also learned throughout the course the importance of reassessing risks. New vulnerabilities are discovered regularly, and system updates provide new vulnerability possibilities. This is why it is important to have a system that constantly reassesses vulnerabilities within it. These will then need to have their risk and cost determined and added to the others to determine the order of addressing them.

The zero trust security model is an approach to security that assumes no user or system should be trusted. Zero trust is not a single tool, but a system of tools the requires constant authentication and provides only access to tools necessary to complete the task(Brook, n.d.). This goes with the principle of not saving security until last by providing areas of vulnerability to look for. This area is where anything and everything can be a risk. Nothing and no one should be trusted. This is because an attack can come from an insider or an outsider. This is why the concept of the least privilege needs to be able to ensure that even insiders can’t assess all data or systems.

Tying all these principles and strategies together is the implementation of the security policies. This is used to ensure that each developer is accountable to ensure security. They are expected to have security considerations during the design and development phases. Assessments should be made to determine which vulnerabilities need to be addressed. The system should run on the Triple-A principle. The user should require authentication, authorization of the usage, and accounting of the use. This means each user has to prove who they are, that they are able to use the system or data, and that usage should have a log for tracing it. Having a policy that supports these will provide the best foundation for secure code and systems. It combines multiple strategies and has multiple layers of security.

Overall, security standards and implementations are a must. It helps to keep users secure and prevents misuse of a system. By understanding different principles and standards, a robust security system can be created. These topics will help me to apply this in my future coding and career. It shows the importance of security and why there are so many fields focused on it.

Reference

Chris Brook. (n.d.). Digital Guardian. *Understanding the Zero Trust security model to safeguard digital infrastructure*. [https://www.digitalguardian.com/blog/understanding-zero-trust-security- model-safeguard-digital-infrastructure](https://www.digitalguardian.com/blog/understanding-zero-trust-security-%09model-safeguard-digital-infrastructure)

Seacord, R. C. (2013). *Secure coding in C and C++* (2nd ed.). Pearson Technology Group.