Kelsey Wanderlingh

SNHU CS 405

October 24, 2025

Module 8 Journal: Reflection

In this course, I have learned a lot about security. Not just about how to code in a secure way, but how to think about and incorporate security throughout the entire software development lifecycle. It starts before development even begins and continues even after deployment. One of the first things the team should do is set up a security policy that all developers should follow. The security policy will help keep everyone accountable by defining what is required of the team from a security perspective (Dunham, 2020). Having guidelines to follow and adhere to throughout the process keeps the code standardized, making it easier to debug and analyze between developers. It is also helpful to have a set of guidelines available to view at any time all in one place. Another good thing about establishing a security policy early is that it requires the team to think about the risks and costs of mitigation. What security layers are mandatory, and what security layers would result in too much time or money loss? Figuring this out early before development on some of the more expensive layers begins can save the team a lot of time.

A secure coding standard goes along with the security policy and contributes to the standardization of written code amongst different developers. A coding standard that implements important rules for avoiding common errors, like not clearing memory when it is no longer in use, can save the team a lot of trouble and prevent crashes that may lead to undefined behavior. Something like the CERT coding standard can be used to implement the required standards for

the job, and those can be added to the security policy. This helps keep the whole team in the know. These standards combined with other forms of security like static code analysis and unit test can detect these errors throughout development and prevent them from ever making it to production.

Outside of development is another important concept known as zero trust. This refers to the idea of "requiring verification of all internal and external access attempts, eliminating trust to prevent breaches" (Brook, 2024). Sometimes, not even internal forces can be trusted. Nowadays it is too easy to do harm from within the system, whether access was obtained through some sort of attack or not. With this model, users will need to continuously verify their identity. They will also lose the privileges that are not 100% needed for their task. This kind of security model goes along with the principle of least privilege and default deny. Instead of only trusting who you absolutely know, just assume everyone has bad intentions, even those who might otherwise be trusted. This is because, even if they *are* trusted, access to their system could be compromised. This is why the zero-trust model requires continuous verification and authentication for every connection attempt. Taking every bit of precaution is necessary in a world where hackers are continuing to develop and enhance their attacks.

References

Brook, C. (2024). *Understanding the Zero Trust Security Model to Safeguard Digital Infrastructure*. Digitalguardian.com. https://www.digitalguardian.com/blog/understanding-zero-trust-security-model-safeguard-digital-infrastructure

Dunham, R. (2020, May 5). *Information Security Policies: Why They Are Important to Your Organization*. Linford & Company LLP. https://linfordco.com/blog/information-security-policies/