Journal: Final Reflection

Throughout this class we examined the adoption of a secure coding standard, by applying security at multiple levels of the system, including during development. At a systems level we examined Defense in Depth, and what it looks like to create a multi-layered security plan that protects the system from threats, vulnerabilities, and potential attacks in multiple, overlapping ways. We examined what it looks like to have layers such as a firewall, encryption, authentication, authorization, anti-virus software, anti-malware software, and secure coding practices all working together to mitigate risks that the system may face.

We spent considerable time with learning to code using secure patterns so that secure patterns such as preventing SQL Injection, implementing Unit Testing, implementing try-catch blocks to handle all exceptions, XOR encryption and decryption, and other secure patterns could all be added to a security policy and be used to code securely during the development process and not as an afterthought. I am currently working as a software engineer, and this class has reinforced to me how important it is to think about security throughout the development process. It is so important when designing new software to add in security to the design and code with these secure coding patterns so that things like secure coding patterns, encryption, the triple-As of authentication, authorization, and authoring, and being able to follow principles such as the principle of least privilege can be followed.

I also learned about the principle of Zero Trust, and how it looks to implement this policy, and that it’s not just users that shouldn’t be trusted without being verified, but also other services, data that is coming in, and any entity trying to access the system. It really broadened my view of how to code securely, even with things like a service-to-service call, or a service (as opposed to a user) inputting data and how this still needs to be sanitized and validated before moving to another system as this service that is inputting the data is not trustworthy as the default.

I also really enjoyed looking at all the different vulnerabilities and non-compliant versus compliant coding patterns that were suggested when looking at implementing secure coding policies. Categorizing based on likelihood and impact and making sure that these things are prioritized and planned on being fixed as soon as possible should be made a priority. In my company, we talk often about “tech debt” and how we need to make sure that we are working to eliminate tech debt, and this class has really made me think about how I can code to keep my tech debt footprint that I leave behind to a minimum, and how as I go and work on different stories and projects for my company, that I should be looking at areas where security could be added or coded by any fixes that I am currently doing to help alleviate some of that tech debt before it becomes a security risk or hole.