### Southern New Hampshire University

### Module 8-2 Assignment: Journal

Daniel Pham

### Professor Demory

### CS-405 – T2851

December 18, 2022

**Adoption of a secure coding standard, and not leaving security to the end**

Over the course of many classes in my journey to obtain my degree in Computer Science with a concentration in Software Engineering, my views on security and its significance towards software development have changed significantly. For example, one of the first classes that every student takes at the beginning of this degree plan is CS-250, “Software Development Lifecycle” (SDLC). One of the objectives in this class was to learn and apply the different stages of the SDLC in an agile development environment. One of the ideas that I took away from this class was that every component in an application should have their own specialty team dedicated to perform each and every task. At the time, I completely disregarded the importance of security in an application and completely disregarded its significance until the final stages of testing and integration. As I gradually progressed through my degree plan, the classes taken had a heavy emphasis on security and naturally, I picked up on the more advanced methods of preventing threats as well. CS-305 and CS-255, “Software Security” and “System Analysis and Design” respectively, are the classes that I took at the same time and realized that secure coding practices can be incorporated into any application at the start of the development process.

**Evaluation and assessment of risk and cost benefit of mitigation**

The factors taken into consideration when prioritizing the standards are its “Severity” which is how impactful the problem will be if action isn’t taken, “Likelihood” which is how likely the problem is to occur, and “Remediation Cost” which is how much resources it will cost to repair the problem if it were to occur. The solution to preventing a high remediation cost is to act on the threat as soon as possible. There’s possibly a large requirement of resources to enforce security best practices in the initial stages of software development but it may be better than waiting until a threat occurs because overall cost may end up costing more than if the DevSecOps were incorporated initially.

**Zero trust**

The zero-trust policy means that treat all activities and users currently on the application's network as potential threats because things may not always be as they seem. Earlier in the term, one of the security best practices was "Don't leave security until the end". Between the two security policies, it seems that the core philosophy of the security best practices is to anticipate the worst possible scenario. Although all of the security principles and best practices can be followed, it does not mean that an application is invulnerable to an attack. Every step of the development process should have security as top priority, and there should be tests ran to double check for threats and breaches constantly. A scenario of "Zero trust" being tied in is assuming that the rest of your team intentionally did not correctly secure the application and it is up to you to fix the mistake. Although assuming that your fellow developers could potentially have ill intent can be frowned upon, the argument can be made that everyone must be cautious as it's not just you double checking the work of your peers but they are also keeping you in check. The idea of every team member keeping each other in check by verifying each other's work can be used to persuade developers who oppose "zero trust".

**Implementation and recommendations of security policies**

I want to conclude this journal entry with emphasizing how important the security process of development is. Defense in Depth is a multilayered approach to security that is layered to where different mechanisms can step up if another fails. DevSecOps is the implementation of application security earlier in the software development lifecycle. Default deny is one of the ten secure coding principles and it is one of the most simple to enforce.