Module 3 Journal: DevSecOps

Developers inherently play a huge role in developing secure software. While traditionally, security is not really evaluated until around the time of testing in the software development lifecycle, this may no longer the be best practice when it comes to building inherently safe software (Jeganathan, 2019). I have a background in chemical engineering, so I find it easy to compare software development to chemical process design. If a project manager is designing a system for handling hydrochloric acid (HCl; can be harmful or lethal if a release occurs), the system will be safest if building materials are selected that will not erode in the presence of the acid during the requirements specification stage. On the other hand, if the system safety is not evaluated until after the “design” and “build” stages of the project, they either will not be able make changes to material of construction, or they will need to greatly increase the budget and timeline of the project to go back and make the system inherently safer. In the same way, if software security is not considered until late in the game, it can be extremely expensive to go back and revise to make the code more inherently safe. From my time in industry, upper management teams are resistant to extend deadlines or increase budget for any reason.

As a developer, it is imperative to design inherently safe programs by writing code to mitigate against vulnerabilities through access control, input validation, building high quality code, and other secure practices. To transform a DevOps cycle into DevSecOps, the first step is to add security considerations to each step of the development cycle. This could look like determining security requirements during the requirements analysis, creating security-centric designs, building secure programs based on the design, and testing and evaluating system security during the testing phase. During the product release, permissions are configured to ensure security infrastructure works as planned, for example, implementing permissions with principle of least privilege. Last, as the system is being used, security should still be considered as security updates for dependencies come out and as data is collected on security-related incidents (Jeganathan, 2019). This will not only produce more secure code, but also likely save time over the course of the project because major security revisions will occur far less often in the later stages of the project. I recommend implementing this continuous security consideration into project lifecycles because it creates room for making more advantageous security decisions from the beginning of the project through to completion/operations.

# References

Jeganathan, S. (2019). *DevSecOps A Systemic Approach for Secure Software.* ISSA Journal.