**CS 305 Module Three Journal  
Alexander Freeders**

As a developer, my role in solving security concerns spans all of the software development lifecycle. In moving up and down the stack, there is no spot where security is *not* a concern. With that being said, security is both a proactive and reactive system, as we do all we can to ensure that security is created securely and maintained (proactive), however when vulnerabilities are discovered and exploited, we must react and work quickly to patch said vulnerabilities (reactive).

Solving security concerns as a developer can look like a variety of things based on what type of security you are implementing, such as network, cloud or application security. A lot of security issues come from poorly written code, insecure storage, and lack of active updating / maintenance (Forsbak, 2024). Solving these issues means identifying threats, mitigating potential threats and continuously ensuring software is up to date and there are no vulnerabilities present.

Security falls at every level of the stock and SDLC. From input validation, storage encryption, Patch management, hardware considerations, cloud configurations, all aspects of the stack require security to be minded. Likewise, in the SDLC, when planning (defining requirements), designing (apply secure principles, plan for secure data flow), implementing (Static Analysis) and testing (Penetration testing), we should be mindful of security concerns as well (PaloAlto, 2025).

Utilizing the phases of DevOps we can add to each phase in order to transition to a DevSecOps pipeline. Fundamentally, it is important that teams adopt a security-first mindset and undergo training and education to fully understand threats, tools and practices (PaloAlto, 2025). We can incorporate security requirements into our plans, ensure that we follow secure coding standards, dependency scan our builds, test integrations, and implement security scans, monitoring, logging and reporting.

The article indicated outlines a plan to secure a DevOps pipeline while not affecting the original intent of the pipeline. through risk assessment and early-life threat assessment. Authorization techniques such as MFA are outlined as well as protecting tools and user credentials. The plan also defines protection controls with segregation of duties for live environments. It is highly recommended that this plan be used. Ensuring that security is at the forefront of our projects will ensure that we remain secure and do everything in our power to maintain that secure environment.

**References:**

Forsbak, Ø. (2024). *Top 10 best practices for secure software development*. Top 10 Best Practices for Secure Software Development. https://www.orientsoftware.com/blog/software-development-security/

PaloAlto. (2025a). *How to transition from devops to devsecops*. Palo Alto Networks. https://www.paloaltonetworks.com/cyberpedia/devops-to-devsecops

PaloAlto. (2025b). *What is Secure Software Development Lifecycle (secure SDLC)?*. Palo Alto Networks. https://www.paloaltonetworks.com/cyberpedia/what-is-secure-software-development-lifecycle