3-2 Journal: Reflection

Sean M Jetté

CS-305 Software Security

Professor Norman

* ***What is your role in solving security concerns as a developer? What might solving security concerns as a developer involve?***

As a developer in the software development lifecycle (SDLC), it is critical to make sure that security issues are regularly resolved by putting best practices into place to guard against vulnerabilities while keeping up with the most recent developments to prevent or reduce them. According to an article in Gitlab (2023b), “security was part of a separate organization known to swoop in after the code was committed, find security issues, and demand changes from (perhaps not surprisingly) reluctant developers who’d already moved on to the next project.” In the modern landscape of programming, security needs to be implemented throughout the development process and not just some afterthought. These practices might include, but are not limited to, making sure the code base complies with practices like input validation or sanitization and even secure coding guidelines recommended by groups like OWASP; as well as creating threat modeling that examines security designs, business requirements, and vulnerability and penetration testing (Beaver, 2019). To discover and reduce risks, developers must also keep up with the most recent vulnerabilities and utilize tools like Maven dependency check plugins along with static analysis tools of the codebase. Furthermore, solving security concerns means collaborating with other teams, testing for vulnerabilities, and continuously improving the application’s security posture. This is not limited to one phase of development but applies throughout the entire software stack, from front-end interfaces to back-end systems.

* ***Where does security fall within the software stack and development life cycle?***

As the SDLC provides a structure for the development of software, implementing security from the start is often a requirement in today’s landscape of threats and vulnerabilities. As an example, during the 1970s when the usage of the Waterfall methodology in software development was in place, revisiting areas of the development process was often difficult due to the sequential nature of such practice. And as such, implementing security change requirements after identification can often be difficult or cumbersome for the development team; ultimately having to restart over from scratch. Thus, as companies have moved on to the Agile methodology, the utilization of sprints for development provides easier processes to inject security practices without having to redo specific areas of the development process. While in the 1970s, security vulnerabilities often resided in backend systems and now with API technologies, threats reside in all functions of the codebase (Snyk, 2020).

* ***How might you add security measures to transform a DevOps pipeline into a DevSecOps pipeline?***

The benefits of transforming from DevOps, with post development security focus, to DevSecOps, that incorporates security at all phases of the development cycle, is that now having security at the forefront of the development cycle helps to not only reduce threat landscape through prioritization but create a culture where threats are a part of the software developer’s toolkit (GitLab, 2023a). Understanding that while DevOps was a continuously evolving function within software engineering, DevSecOps is too, and as the “practices mature, the related tooling, governance processes, developer awareness, knowledge and training need to be updated often” (Nagarajan, 2020). Therefore, for this to be fully transformed, all members of the SDLC team need to have security in mind. This can be manifested in tools such as automated security testing or scanning dependencies throughout the process. Software engineers or developers within the Scrum team need to ensure they focus on best security practices, employ static code analysis methodologies and even integrating vulnerability management when issues arise. By making it everyone’s responsibility, team members at all levels of the SDLC will develop the requisite skillset beyond just software development to a security infused mindset.

* ***The article suggests creating and following a plan to secure the entire DevOps life cycle. What is included in the suggested plan? Would you recommend following the plan?***

Within the article by Jeganathan (2019), *DevSecOps: A Systematic Approach For Secure Software Development*, the plan contains stages for security requirements during the DevOps life cycle. Starting with “high-level rapid risk assessment for the new release and quantify the risks by evaluating the threat models.” This sentence speaks for itself and is a foundational step for understanding any risks through threat modeling—leading me to think it would be very much recommended within my own DevSecOps implementation. Next, the plan discusses using a secure method for web lifecycle tools by giving examples of Gitlab and Azure DevOps. These options are great for development due to several reasons—where security is already practically built in with options like Gitlab; especially if you’re looking for a comprehensive DevSecOps solution. If you’re already in the Microsoft ecosystem, Azure DevOps, which now integrates with GitHub, can be a great option for accessibility, scalability, security, and cost efficiency; as with cloud-based services, upkeep for maintaining your own servers can significantly increase costs (Microsoft, 2024). The next part of the plan is ensuring user access keys with privilege service accounts, API keys, and lastly protection in mind while also defining infrastructure protection controls and enforcing of segregation duties. Given the complex nature of modern security threats, the plan’s suggestions are excellent starting points for incorporating security into the DevOps lifecycle. While security practices can always be improved, this approach provides a solid foundation for embedding security within the development process.

References:

Beaver, K. (2019, March 4). *The developer’s role in application security strategy*. Search Security. https://www.techtarget.com/searchsecurity/tip/The-developers-role-in-application-security-strategy

GitLab. (2023a, April 10). *What is DevSecOps? | GitLab*. GitLab. https://about.gitlab.com/topics/devsecops/

GitLab. (2023b, April 18). *What is developer-first security? | GitLab*. GitLab. https://about.gitlab.com/topics/devsecops/what-is-developer-first-security/

Jeganathan, S. (2019, November). *DevSecOPS: A Systemic approach for secure software development.: Multi-Search*. https://eds.p.ebscohost.com/eds/pdfviewer/pdfviewer?vid=1&sid=a043e0cb-fbdb-45d4-82f2-2fc28c519fd0%40redis

Microsoft. (2024, February 1). *What is Azure DevOps? - Azure DevOps*. Microsoft Learn. https://learn.microsoft.com/en-us/azure/devops/user-guide/what-is-azure-devops?view=azure-devops

Nagarajan, D. (2020, December 7). *How to transform from DevOps to DevSecOps*. Security Intelligence. https://securityintelligence.com/posts/how-to-transform-from-devops-to-devsecops/

Snyk. (2020, June 21). *Secure Software Development Lifecycle (SSDLC)*. https://snyk.io/learn/secure-sdlc/