Devin Bashaw

CS 305

01/31/2025

Software security is essential to software development, ensuring applications we design remain safe from attacks and data leaks. Security breaches can lead to many types of losses, from user private information to government facility codes. The entire countries infrastructure as we know it can be damaged from a security attack. Developers are thus responsible for safeguarding these applications and putting protocols into place to keep us safe. Developers have to stay informed about emerging security threats to integrate measures into the software development life cycle, if we do not take proper measures we are all screwed.

Solving security concerns can be a difficult task but luckily is mostly preventable when following protocols. Security needs to be implemented at all levels of the software applications, even in the very beginning of the development process. Vulnerabilities can happen at any stage, and implementing them early ensures that security vulnerabilities are detected and mitigated before they become a much bigger problem (DevSecOps: A Systemic Approach for Secure Software Development).An attack can happen anywhere in the application: api, database, network, and UI, so these layers need to be retroacvtively protected with routine testing, control mechanisms and monitoring (Secure Coding Guidelines for Java SE).

DevOps is a set of tools used by software developers to accelerate development through routine integration and delivery, otherwise known as CI/CD. DevSecOps is used to make DevOps better by implementing the security DevOps is missing (DevSecOps: A Systemic Approach for Secure Software Development). Security is implemented at every stage of the pipeline through automated scanning, security as code, awareness training, monitoring, incident response, threat modeling, and risk management. Automated scanning implements static application security testing, or SAST, and dynamic testing DAST to find early stage vulnerabilities. Security as code means embedding policies within the infrastructure as code (hardcode the security in essentially) to ensure compliance is met. Awareness training ensures all developers on the team are taught these security protocols and safe coding practices (unit testing, try except, tokens). Monitoring can be implemented with the usage of tools or dependencies, same with incident responses (asserts). Lastly, threat modeling and risk management can be used to identify and mitigate threats to the software before anything bad actually happens. If these measures are implemented, then the DevOps pipeline can transform into the DevSecOps pipeline.

The plan includes these steps: plan, code, build, test, release and deploy, and operate. In short, conduct risk assessments and define security policies, implement contro mechanisms, write secure code and monitor it, use tools and dependencies to detect vulnerabilities, test the code and run security scans, ensure the code meets compliance requirements and safe coding practices, automate security validation, manage configurations, secure containers, networks and database, and then once released continuously monitor for threats, run routine checks, updates, and automate an incident response (DevSecOps: A Systemic Approach for Secure Software Development). Regarding this plan and whether I would recommend utilizing it, I do recommend utilizing it because it is a pretty safe proof plan for developing software to ensure it is protected. There will only be advancements in the security sector as time goes on, and those advancements should always be implemented. All I can say is to ensure the software NEVER goes outdated.

The role of a developer in software security is critical to the lifespan of the application. One cannot call themselves a developer unless they follow the security protocols and safe coding practices necessary to ensure the software being developed is protected against crime and leakage. By transforming a DevOps pipeline into a DevSecOps pipeline, developers can have their software protected without doing constant work, making changes, and updating. Eventually we will get ot a point to where a software developer barely has to work at all, due to automation, and we will just have to oversee and maintain that automation instead of implementing anything ourselves.

Works Cited:

* Jeganathan, S. (2019). DevSecOps – A Systemic Approach for Secure Software Development. *ISSA Journal.*
* Oracle. (2025). Secure Coding Guidelines for Java SE.