Module 3: Journal

Software Security

Version 1.4

Ronny Z. Valtonen

Software Security

After reviewing the code workspace of the project, software security takes a lot of thinking and review. Creating a modern web application, it is my role as a software security programmer to solve security concerns by knowing the common areas of attack as well as learning new strategies that attackers use to attempt to bypass security solutions. Solving security concerns as a developer means adhering to secure coding practices. For example, I need to implement input validation and use safe logging for output handling for sensitive information. The Iron-Clad Java textbook discusses techniques to mitigate these types of vulnerabilities that attackers could potentially exploit.[1]

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

Ha, this sounds like a trick question - software security does not fall into a single level of the development life cycle as it is in every level of a software stack and in every part of the development life cycle. Not only does the back-end codebase need to be reviewed for software security, the model (middle-man code) and the view (user access / GUI) all need to be reviewed for security vulnerabilities. During the development process, developers need to gather information on what the purpose of the software stack is and consider potential threats and security policies, implementing secure frameworks as the software is being developed. The view also needs to be reviewed as any user input fed through the GUI and into the model to the controller(s) need to be sanitized to prevent injection attacks. So within every part of the development life cycle, software security needs to be a concern.

To add security measures to transform a DevOps pipeline into a DevSecOps pipeline, I would implement automatic Maven Dependency checking at the end of each daily commit or version update. This would ensure that any new / current dependencies do not have vulnerabilities. Furthermore, I would implement automatic security tests, such as known injection techniques for any user input to make sure all user input is sanitized (though I don't know how to do this yet). Lastly, I would recommend using plants

[1] Manico and Detlefsen, *Iron-Clad Java: Building Secure Web Applications (Oracle Press)*.

for any attacks that do inevitably fall through (no company is 100% protected from all attacks), to ensure that vulnerable data is protected and provides accountability to continuous improvement in security practices for the company.

After reviewing the article written by Seetharamn Jeganathon on “A Systemic Approach for Secure Software Development”, it is suggested to follow a comprehensive plan to secure the entire DevOps cycle.[2] The plan recommends performing a high-level risk assessment for every new release, using common DevOps lifecycle tools like GitLab and Azure, and defining protection measures in-place. All of this also includes continuous integration, continuous delivery and continuous security. As I discussed previously to answer the question of where security measures fall in place, security needs to be continuously integrated. Meaning that I would recommend following the plan while enforcing coding practices like OWASP for dependency vulnerability checking and security testing.

[2] Jeganathan, Seetharaman. “A Systemic Approach for Secure Software Development.” *ISSA*, November 2019.