NIST Position Paper: Area #4

GitLab comments on the *Initial minimum requirements for testing software source code* in support of the May 12th 2021 Executive Order on Improving the Nation's Cybersecurity, drafted May 26th, 2021.

> *Initial minimum requirements for testing software source code* including defining types of manual or automated testing (such as code review tools, static and dynamic analysis, software composition tools, and penetration testing), their recommended uses, best practices, and setting realistic expectations for security benefits. See EO Sections 4(e)(iv and v) and 4(r).

GitLab is excited to partner with NIST and other federal cyber stakeholders on Standards and Guidelines to Enhance Software Supply Chain Security. As a company that holds transparency as a core value, we are happy to share our learned best software security practices to help influence standards that strengthen the cyber resiliency of federal agencies and the broader cyber community.

Testing software for vulnerabilities should begin early in the development process as a general guidance. Threat modeling should be performed prior to or during the development phase and as new features or components are added to the software. Full threat modeling is not required for all software development, but using an evidence-driven threat model by developers, prior to development, can determine required best practices.

Best Practices:

Source code and software should be assessed using the following methods:

I. Vulnerability management
   a. Static Analysis
   b. Dynamic Analysis
   c. Container scanning
II. Secrets management
   a. Secrets scanning
III. Dependency management
   a. Dependency Scanning/Software Composition analysis
IV. License Management
   a. Open-Source Licensing scanning
V. Code review
VI. Penetration testing (where applicable)

The above security methodology should be applied upon source code. Scans should be performed continuously throughout development and throughout the operational life of the software.

Recommended uses:

Static analysis should be used during the build to identify security weaknesses in the application data and control paths. This not only allows for quicker identification of weaknesses in the software but also identification at a precise location. Dynamic analysis should also occur during the build phase on executable code to determine vulnerabilities.

Secrets scanning is critical to the identification of various keys, tokens, certificates, credentials and passwords which are often exposed in the development lifecycle. The exposure of such secrets has led to the compromise of confidential information, customer data and even large-scale breaches.

Dependency scanning is critical to effective supply chain security. This should occur during the build to identify known vulnerabilities in external dependencies and libraries.

Open-Source license scanning can ensure compliance with appropriate policies and legal requirements.

Code review provides a final opportunity to detect insecure and poorly written code as well as misconfigurations, hardcoded secrets and general software vulnerabilities.

Penetration testing should occur on the initial software build as well as at least once annually by a qualified, independent third party or qualified internal team.

Realistic expectations for security benefits:

Should a vulnerability be discovered, an assessment of severity and impact should be applied to determine remediation efforts, mitigation techniques and or risk acceptance. A trust relationship has to be built in order for a collaborative approach towards mitigating the problem.

It is recommended vulnerabilities and dependencies are managed through a well-defined and established process that maintains monitoring, tracking and remediation standards.

The expectation is not to introduce significant friction and slowdown to the development of software but to ensure a secure, efficient and faster devops pipeline. The early identification and response to vulnerabilities and supply chain weaknesses will significantly improve the security of an organization's development lifecycle.

In Closing

GitLab appreciates the opportunity to offer these positions to NIST in relation to EO 14028. We have proposed approaches that have proven to be fruitful in the modern software delivery model.

Submitter:

- Johnathan Hunt
- VP of Security, GitLab
- jhunt@gitlab.com