VENAFI VERACODE

Securing Modern Software Development Pipelines

Written by: Shivajee Samdarshi, VP Engineering, Venafi (www.venafi.com)
and Chris Wysopal, CTO and Founder, Veracode (www.veracode.com)

Software security leaders, Venafi and Veracode, are calling for fundamental changes in how organizations secure their software development pipelines. Recent attacks, such as those on SolarWinds, Codecov and others have illustrated that the world's modern software supply chain—including development, build and delivery—is vulnerable to attackers.

We are addressing NIST's position statement area: "2) Initial list of secure software development lifecycle standards, best practices and other guidelines acceptable for the development of software for purchase by the federal government."

Software development has changed. Technologies such as cloud-native, application containers, open source and DevOps have transformed how our organizations develop software. This change has been occurring for years.

Unfortunately, information security has not kept up with the rate of change in software development. In most organizations, InfoSec still operates largely isolated from software development teams. This results in a siloed organization where development teams and InfoSec teams often do not collaborate with each other, instead looking at the other as the adversary rather than a partner.

A New Approach Is Needed

Our position is that a new approach and point of view is needed to secure the software development pipeline. Furthermore, we believe this problem must be addressed by the engineers developing their software rather than InfoSec. No single security measure will prevent future incidents. Instead, multiple measures throughout the software development lifecycle must be taken.

We propose a standard set of controls[1] to secure the software development pipeline for continuous integration and continuous deployment (CI/CD) against attack. The goal is to minimize the possibility of a supply chain attack by ensuring that authentication and authorization is properly managed throughout the pipeline, the integrity of software artifacts are tested at appropriate stages and controls are placed on third party and open source software incorporated into the software.

---

[1] https://github.com/Venafi/blueprint-securesoftwarepipeline - written by Venafi and Veracode with contributions from CloudBees and Sophos

We base our approach on the following design philosophy:

- **Least Privilege** - grant only the access and permissions required to accomplish a job.
- **Immutability** - Artifacts and infrastructure are not modified after deployment to an environment. If a change is required, it is done in the image or script within the development environment and then promoted through the higher environments.
- **Everything as Code** - Infrastructure, security policies and other parts of the pipeline are implemented as code and subject to the same controls as the software artifacts.
- **Traceability** – All changes—whether to infrastructure or business code—is revision controlled. This principle works hand in hand with Everything as Code.

The design is pragmatic. Security controls will not be implemented or will be routed around if they prove an impediment to the timely delivery of new software.

Our approach is designed on the four stages of software development pipelines:

- **Code**: developers design software and commit code to code repositories
- **Collaborate:** developers include external and internal libraries and share software for review
- **Staging**: software is built and prepared for final delivery
- **Production**: software is run anywhere

Multiple security measures must be built into each of these stages. Security measures include, but are not limited to:

- Restriction of administrative access to CI/CD tools
- Restrict usage of intermediate software artifacts (e.g., source code) to only those that have been signed with a developer GPG key
- Ensure that access keys (e.g. TLS certificates, code signing keys, etc) expire automatically in a shorter period of time
- Restrict usage of software dependencies to only trusted registries
- All artifacts needed for a software build are stored in a repository
- Before any artifact is used, its digest should be validated against the repository to ensure it has not been compromised
- A pull request requires multiple reviewers
- Critical artifacts are code signed and those signatures validated
- Scan deployed images in production
- Ensure build environments are ephemeral and immutable

To summarize, it is our position that the industry needs to rethink how software development pipelines are secured and must incorporate security measures throughout the pipeline. A single security measure is not adequate. This problem must be owned by the engineering teams who use these pipelines in consultation with InfoSec who can help inform them on adequate security policy.