Broadcom and Symantec (A Division of Broadcom) Position Paper on Standards and Guidelines to Enhance Software Supply Chain Security

Issue # 5: *Guidelines for software integrity chains and provenance.*

Author: Amarendra Godbole
Title: Technical Director
Email: amarendra.godbole@broadcom.com

Introduction
A software supply chain is the entire sequence of events that happen to software from the point of origin (say, a developer's system) to the point of deployment (cloud service or enterprise customer locations). Each sequence or the element of this chain translates the software in some manner and forms the "circle of trust." Developer systems, build/CI systems, third party software repositories, signing keys, compilers, and download portals are some examples of the elements of this supply chain.

The ultimate goal of a trustworthy software supply chain is to keep this "circle of trust" as small as possible, and resist random and unpredictable changes in this circle. The final software artifact is only as trustworthy as the processes, people and tools involved in its development. This document meditates over a set of best practices that help secure this software supply chain to produce "trustworthy" software.

Best Practices
1. Third party software repositories
    - Use only well-known and trustworthy repositories that typically require the original author to sign the software. For example, Maven Central requires PGP signatures.
    - Setup an internal repository cache and point all build systems to fetch from this cache. This can be achieved by restricting access to the internet-based repositories, or resolving DNS requests to the internal repository cache system. This allows for regular scan of malicious software and integrity verification checks on the downloaded software.
    - Obtain third party software over a secure channel (examples: https, sftp, scp)
    - Track all usage of third party software

2. Build and CI/CD systems
    - Any compilation and build of production builds must happen only on designated build systems.
    - Obtain all build toolchain software from trusted sources. This includes (but is not limited to) operating systems, compilers, linkers, as well as integration and deployment software.
    - For critical software builds, prefer physical systems instead of virtual machines.

3. Credential and key management
    - Make credentials for build systems available just-in-time during the build, and then destroy them once the build is complete.
    - Store credentials separate from build and source code systems, with adequate protection.
    - Ensure credentials do not "leak" into virtual machines that are setup on a temporary basis.
    - Leverage source code control system's facility to isolate credentials (for example, github provides "Environments." See https://docs.github.com/en/actions/reference/environments).

- Rotate credentials and keys on a regular schedule.

4. Source code repository
   - Grant access to authorized users only to the source code they need to do their job (“least privilege”). Wherever possible provide read-only access as opposed to read-write access.
   - Mandate multi-factor authentication.
   - Review and adjust access on a quarterly basis.
   - Maintain an audit trail of all access to the repository.

5. Developer systems
   - Regularly patch and keep up-to-date
   - Install security software (anti-virus) for regular scanning for malware and viruses.
   - Manage remotely via MDM if possible. This reduces risk in the event of theft or loss.

6. Customer downloads
   - Make signed downloads available wherever possible.
   - If signed downloads are not possible, provide SHA2 sums of the downloaded software via a separate channel (separate web-portal, email, etc.).

7. Process
   - Have a robust open source review process within the organization. Leverage license and vulnerability detection tools wherever possible.
   - Train developers to select trustworthy third party libraries, as opposed to the first one they encounter on github.
   - In general, institute robust policies and processes for third party software supply chain integrity covering all the best practices outlined in this document.

Summary
Starting with trustworthy third party software repositories and carefully guarding access to Build and CI/CD systems provides a good starting point to a secure supply-chain, until other practices can be implemented.