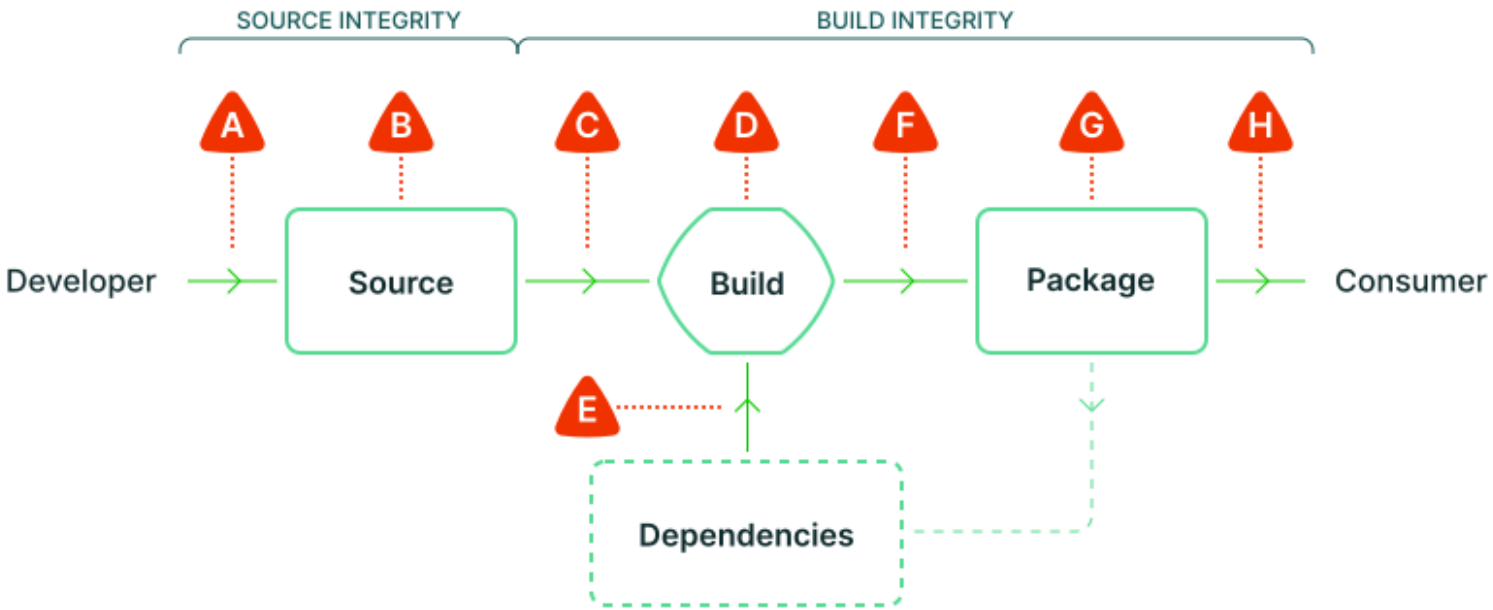
Supply Chain Security

Aleksandr Tserepov-Savolainen

September 29, 2022

Outline

Problem statement



A Submit unauthorized change
B Compromise source repo
C Build from modified source
D Compromise build process
E Use compromised dependency
F Upload modified package
G Compromise package repo
H Use compromised package

Breach cases

TODO: Example breach cases here

SLSA Framework

Source Integrity

Ensuring every change reflects the intent of producer.

SLSA Framework

Source Integrity

Ensuring every change reflects the intent of producer.

Build Integrity

Ensuring artifacts are not modified on transit between stages of the Supply Chain.

SLSA Framework

Source Integrity

Ensuring every change reflects the intent of producer.

Build Integrity

Ensuring artifacts are not modified on transit between stages of the Supply Chain.

Availability

Ensuring that all code and change history are available for potential incident investigation.

Levels of Assurance

Levels of Assurance

Level 1

Easy to adopt, offering supply chain visibility and generating provenance

Levels of Assurance

Level 1

Easy to adopt, offering supply chain visibility and generating provenance

Level 2

Minimal build integrity, minimal SW tampering protection

Levels of Assurance

Level 1

Easy to adopt, offering supply chain visibility and generating provenance

Level 2

Minimal build integrity, minimal SW tampering protection

Level 3

Hardened infrastructure, trust integration

Levels of Assurance

Level 1

Easy to adopt, offering supply chain visibility and generating provenance

Level 2

Minimal build integrity, minimal SW tampering protection

Level 3

Hardened infrastructure, trust integration

Level 4

The highest assurance of build integrity and dependency management

NixOS / Spectrum Build Environment

TODO: Build environment picture
Hydra -> BinCache -> Jenkins -> Release

- — TII GitHub
- — OpenSrc locations

Nix Tooling SLSA Solution

- package signing during build
- Package signing during sharing
- Package signature verification during testing stages

SCA (Software Composition Analysis)

- Automated process that defines the open source software in the codebase.
- Companies need to be aware of potential obligations, limitations and security vulnerabilities that open source brings into play.
- As the codebase grows, tracking all of those becomes rather tricky.
- SCA takes use of automatic scanners to enable productivity without compromise on security.

SBOM (Software Bill of Materials)

- List of software components in a derivation
- Generated at the build time by the build system
- Serves as part of build provenance
- Stored inside of the software package
- Is signed at the build stage

[Vulnix] Theory of operation

- Pulls all known CVEs from NVD
- Matches a list of derivations against CVE entries
- Whitelisting is used to suppress unwanted results