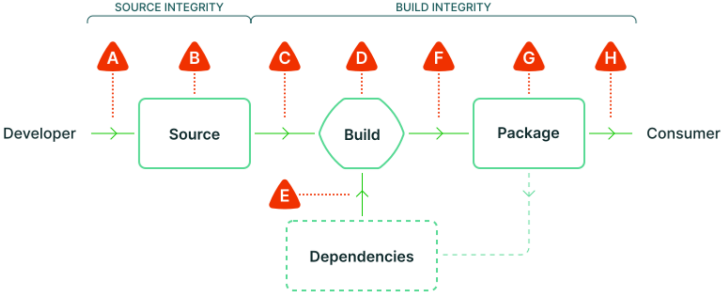
Supply Chain Security

Aleksandr Tserepov-Savolainen

September 23, 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

image src: `https://slsa.dev/spec/v0.1/#supply-chain-threats`

Breach cases

TODO: This one is just an example. Work needed.

1959 CIA intercepted a USSR lunar probe

`https://www.cia.gov/readingroom/collection/lunik-loan-space-age-spy-story`

2014 3rd party vendor credential leak on Home Depot's credit card terminals

`https://www.computerweekly.com/news/2240234281/Home-Depot-traces-credit-card-data-hack-to-supplier-comprom`

2021 backdoor in the open source PHP Git server

`https://news-web.php.net/php.internals/113838`

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

NixOS SLSA Solution

Hydra package signing

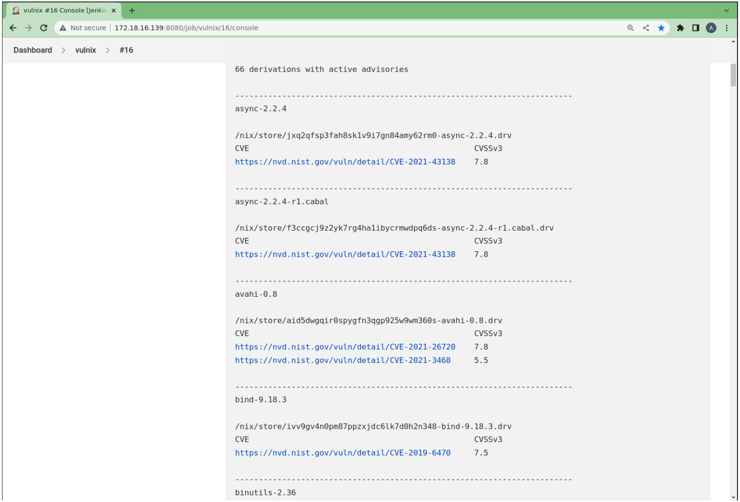
Binary cache package signing

Jenkins package signature verification

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.

Look & Feel


vulnix #16 Console [Jenkin
Not secure
172.18.16.139:8080/job/vulnix/16/console
Dashboard
vulnix
#16

```
66 derivations with active advisories

------------------------------------------------------------------------
async-2.2.4

/nix/store/jxq2qfsp3fah8sk1v9i7gn84amy62rm0-async-2.2.4.drv
CVE                                                CVSSv3
https://nvd.nist.gov/vuln/detail/CVE-2021-43138    7.8

------------------------------------------------------------------------
async-2.2.4-r1.cabal

/nix/store/f3ccgcj9z2yk7rg4ha1ibycrmwdpq6ds-async-2.2.4-r1.cabal.drv
CVE                                                CVSSv3
https://nvd.nist.gov/vuln/detail/CVE-2021-43138    7.8

------------------------------------------------------------------------
avahi-0.8

/nix/store/aid5dwgqir0spygfn3qgp925w9wm360s-avahi-0.8.drv
CVE                                                CVSSv3
https://nvd.nist.gov/vuln/detail/CVE-2021-26720    7.8
https://nvd.nist.gov/vuln/detail/CVE-2021-3468     5.5

------------------------------------------------------------------------
bind-9.18.3

/nix/store/ivv9gv4n0pm87ppzxjdc6lk7d0h2n348-bind-9.18.3.drv
CVE                                                CVSSv3
https://nvd.nist.gov/vuln/detail/CVE-2019-6470     7.5

------------------------------------------------------------------------
binutils-2.36
```

[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

[Vulnix] Pros & Cons

[Vulnix] Pros & Cons

Pros

- Fast
- Easy integration
- Written in Python - easy to maintain

[Vulnix] Pros & Cons

Pros

- Fast
- Easy integration
- Written in Python - easy to maintain

Cons

- Simplistic mapping can lead to false positives / negatives
- Inactive development