THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
NORTH AMERICA

Embedded Linux Conference

Is Maven safe for production?

Adam Kaplan, Red Hat
Manfred Moser, Chainguard

About the speakers

Adam Kaplan
Sr. Principal Software Engineer
Red Hat

- CNCF contributor and maintainer
- Container build and secure supply chain
- Helped ship freight containers
- Swam with Olympians

Manfred Moser
Sr. Principal DevRel Engineer
Chainguard

- Chainguard Libraries for Java champion
- Maven trainer and commiter
- Repository manager expert
- Book author and writer
- Open source hacker

Agenda

- What is “safe” and “for production?”
- What is Maven?
- “Battle stories”
    - Generating SBOMs
    - Harden builds
    - SLSA
- How can we improve security?
- Q&A

What does “safe” and “for production” even mean?

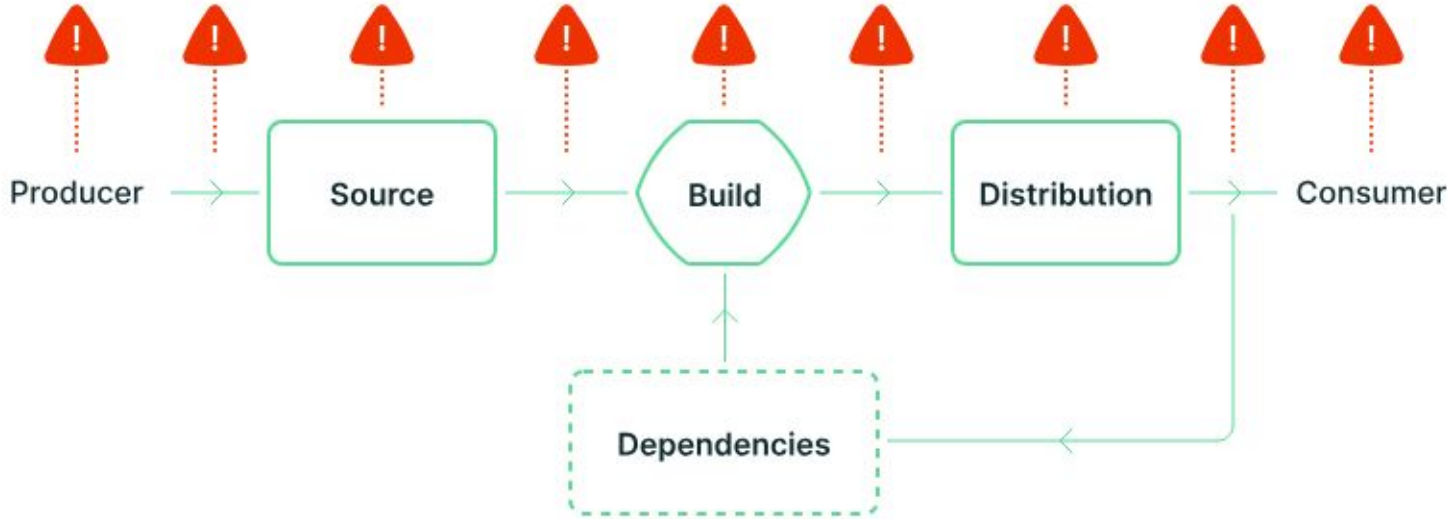
Quality and security of your software supply chain

*It should be **secure** to run a **reproducible** build and **reliably build** your application:*

- Maven, plugins, extensions, repository managers
- External sources - Maven Central and other repositories
- Build runtime environment
- Output artifacts and complementary info

Classification as **S**upply-chain **L**evels for **S**oftware **A**rtifacts - https://slsa.dev

SLSA threat model - attack vectors everywhere



What is Maven?

Maven build ingredients

- Mostly widely used Java/JVM build system
- A flexible plugin execution framework
- Supports whatever you can implement in a plugin
- Nearly all functionality comes from plugins
- Strict around convention over configuration

It just works

mvn clean install

Plugins and extensions

- Core building blocks for functionality
- Plugin version are set in Maven binary
- Often changed in organization or project (pom.xml)
- Loaded from repositories like other dependencies
    - quickstart example - two project dependencies, >400 dependencies overall
- Extensions: customize Maven, some very useful features possible
    - Polyglot Maven
    - Mimir

Dependencies

- Everything your **app** and **build** need
  - Scopes - compile, test, runtime
  - Including plugins
- Plugins and extensions have dependencies
- Dependencies have dependencies
- Automatically resolved to one version
- All are sourced from repositories
- Maven Central by default

```
<dependency>
  <groupId><groupId>
  <artifactId></artifactId>
  <version></version>
  <classifier></classifier>
  <scope></scope>
  <type></type>
<dependency>
```

Security of maven central repository

- **Strict user validation**
- **Required** package namespacing (group ID)
- **Verified** namespace access from DNS and other sources
- Artifact checksums and PGP signatures
- Verification during staging, before publication
- Malware prevention, using signals from Sonatype Nexus Firewall

Operated by Sonatype - https://central.sonatype.com/

Battle stories: Software bill of materials

THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
NORTH AMERICA
Embedded Linux Conference

What is an SBOM?

- Software Bill of Materials
- Made for machines
- Standardized formats
- Vendors may need to produce this (ex: CRA)

How can I make an SBOM with Maven?

- Maven dependency plugin:

  ```
  mvn dependency:tree -DoutputType=json
  ```

- CycloneDX Maven Plugin:

  ```
  mvn org.cyclonedx:cyclonedx-maven-plugin:makeAggregateBom
  ```

- SPDX Maven Plugin:

  ```
  mvn org.spdx:spdx-maven-plugin:createSPDX
  ```

SBOM for plugins?

Plugins

- Gap in existing tools
- Dynamic dependency loading
- Non-Java dependencies

Extensions - more challenging!

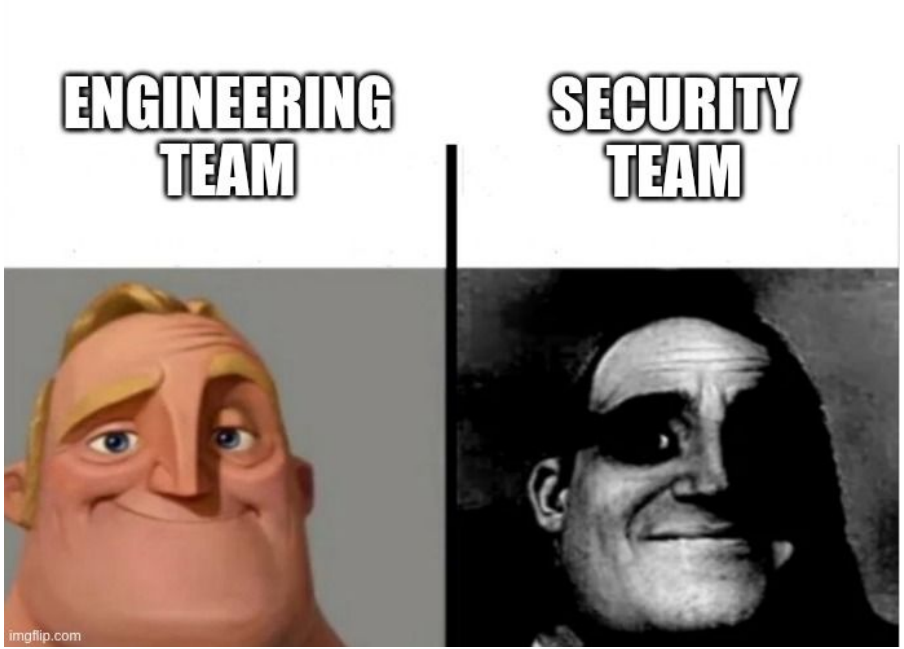
- Core extensions “outside project” - early loading
- Build extensions “inside project” - lazy loading

```
build.plugins[*].plugin.extensions=true
```

```
build.extensions[*].extension
```

Shading

- Use case: “über JARs”
- Dependencies can be included as ***relocated compiled code***
- “Reduced POM” output hides dependency tree!



Battle stories: Hardened builds

THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
NORTH AMERICA
Embedded Linux Conference

Dependency rebuilds

Why do this?

- Prevent malware
- Add additional metadata
    - SBOMs (CycloneDX, SPDX, etc.)
    - Provenance
- Security patching*
- Control of supply chain

Code archeology

Rebuilding old releases is hard:

- Where is the source code? Is the release commit/tag correct?
- Not in a mono repo for mixed use ideally
- Does mvn clean install work?
- Before JDK 8 is harder since no open source distro available

Hope is doesn't use Ant or Gradle…

Obstacles and challenges

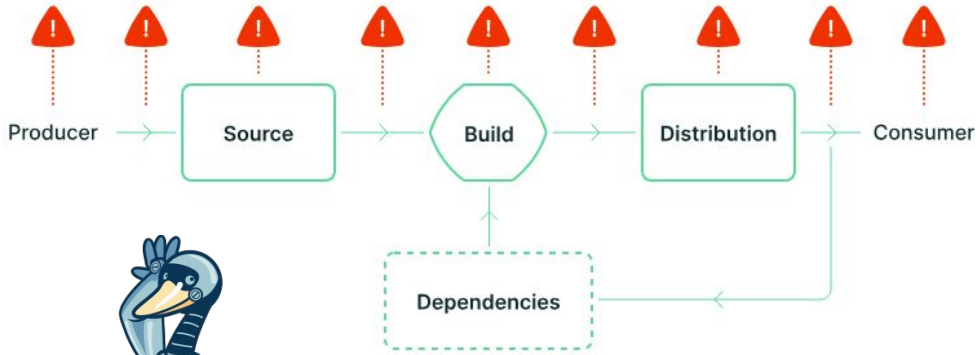
- Different Java language level for different modules
  - Potentially difficult to choose JDK
  - Trino JDBC driver vs Trino itself
- Release frequency can be too high and too low
  - aws-sdk or wso2 - every commit?! - too much churn
  - Apache hive - about once a year or less - too many security issues
- Native code is tricky and platform dependent, luckily rare
  - Tomcat JNI
  - RPM builds from Maven projects
  - Airlift compression library
  - Go-based launcher in Trino

Battle stories:
Secure builds and making SLSA

THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
NORTH AMERICA
Embedded Linux Conference

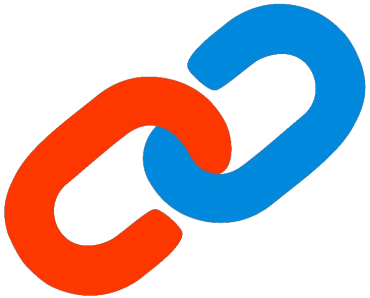
Making SLSA

- **S**ecurity **L**evels for **S**oftware **A**rtifacts
- Key feature: *signed system attestations*
- Published “tracks”
  - **Build** - harden the build infrastructure
  - **Source** - harden the source code
- Not a silver bullet - see “mini Shai-Hulud”



Attestations and signatures

- Attestation: machine readable statement (in-toto)
- Summary attestation (VSA)
- Signing
  - Traditional PGP
  - Cosign “keyless”



in-toto

sigstore

cosign

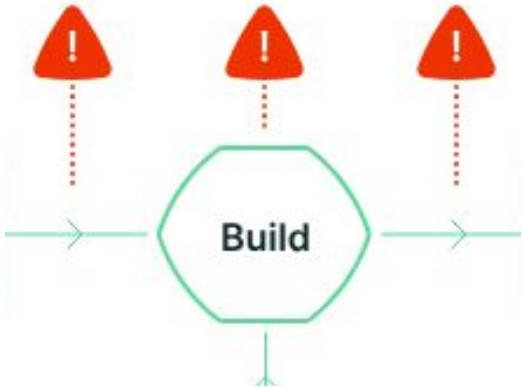
Build threats

Threat: build from bad code, process, system

Mitigation: system-generated attestation

Threat: tampered artifact or attestation

Mitigation: signed artifacts and attestations; digest references



How to generate?

- Build task/step
- “Observer” pattern - Tekton Chains
- GitHub Action Workflow

Source track

Threat: code without context

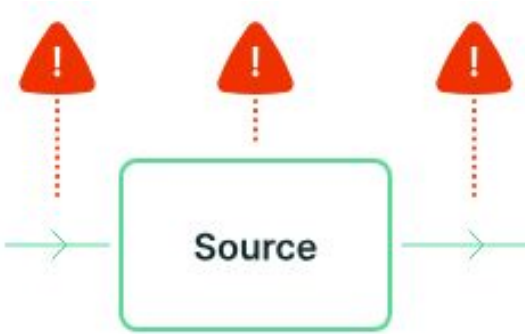
Mitigation: modern version control system

Threat: insider overwrites

Mitigation: branch protection

Threat: unauthorized code change

Mitigation: coherent history, identity management



Threat: privilege abuse

Mitigation: two-party review

Draft tracks

Build environment

- Harden the build machinery
- Attest the execution environment
- Validate expected environment state prior to build execution
- Hardware-level attestations

Dependencies

- Inventory build dependencies
- Vulnerability triage and remediation
- Trusted sourcing of third party artifacts
- Policy-driven dependency consumption

Dependency level 4 and Maven

- Dependency inventory ✅
- Vulnerability triage tools ✅
  - Renovate
  - GitHub Dependabot
- Producer-controlled dependency sources ✅
  - Mirroring encouraged
  - Strong vendor ecosystem
- Proactive defense from upstream attacks ✅
  - Verified artifact domains (Maven Central)
  - Required signatures and sources (Maven Central)
  - Immutable versions (Maven Central)

How can we improve safety?

Lock down your Maven use

- Control Maven version with wrapper, standard build env, and enforcer
- Pin your Maven plugin and extension versions
- Ensure correct Java version and config - enforcer plugin, .mvn/jvm.config
- Maven configuration in ./mvn/maven.config
- Potentially standardized settings.xml in .mvn

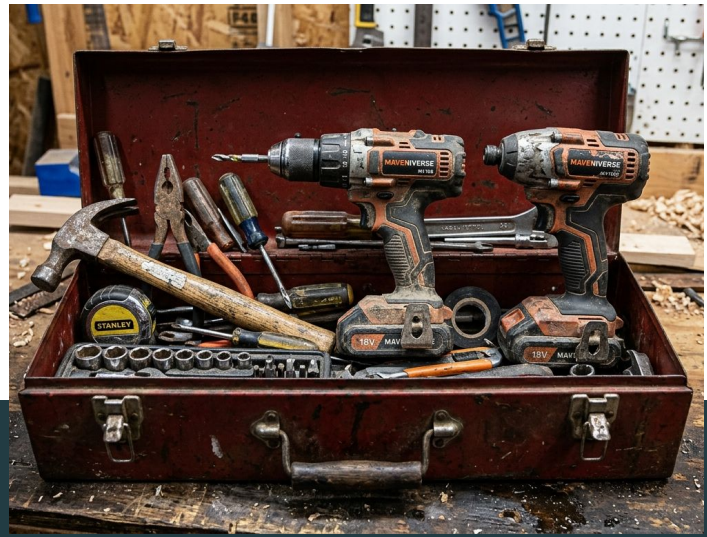
Dependency safety

- Understand your dependencies
  - dependency:tree|analyze|list
- Automate dependency updates
  - Renovate, Dependabot, versions plugin
- Update often
- All are sourced from repositories
- Understand those source repositories
- Use a repository manager as cache and control layer

```
<dependency>
  <groupId><groupId>
  <artifactId></artifactId>
  <version></version>
  <classifier></classifier>
  <scope></scope>
  <type></type>
<dependency>
```

Useful plugins, tools, and tips

- Help, versions, enforcer, and various other plugins
- Maveniverse tools and libraries
  - Njord - deployment to Maven Central
  - Mima and Toolbox
  - Mimir - better local cache
  - Pilot - terminal UI
- Reproducible builds guide

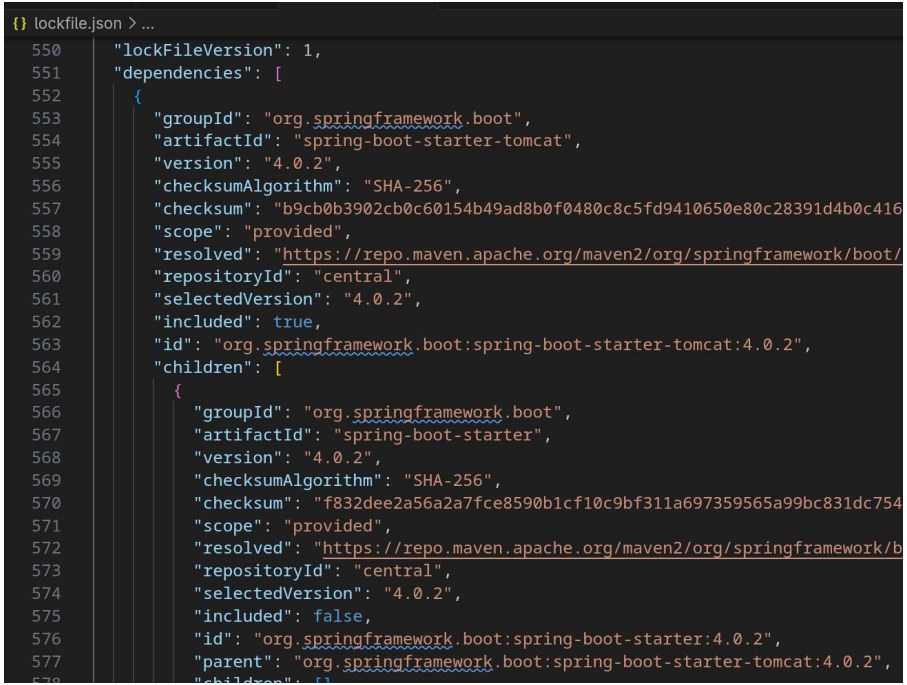


Checksums and lockfiles for Maven

- Trusted checksums

```
-Daether.artifactResolver.postProcessor.trustedChecksums=true
-Daether.artifactResolver.postProcessor.trustedChecksums.checksumAlgorithms=SHA-256
-Daether.artifactResolver.postProcessor.trustedChecksums.record=true
```

- Maven lockfile plugin



OpenSSF Community Day

NORTH AMERICA 2026

Making a Lockfile for Maven

What else can we do?

- README file with build and release profile instruction
- Make sure mvn clean install works
- Don’t use version ranges …
- Make your build independent of a version control system
- Do the same for the release build configuration

THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
NORTH AMERICA

Embedded Linux Conference

Is Maven safe for production?

THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
NORTH AMERICA

Embedded Linux Conference

Is Maven safe for production?

Yes!

Ways to contribute

- Improve your own project
- Follow our advice
- Update dependencies
- Use the tools and thank the maintainers
- Help upstream

References

- Maven: https://maven.apache.org/
- Maven Central: https://central.sonatype.com/
- SLSA: https://slsa.dev
- Maveniverse tools: https://maveniverse.eu/
- Maven build requirements
  https://github.com/simpligility/maven-build-requirements
- Maven Lockfile Plugin: https://github.com/chains-project/maven-lockfile

Questions?

THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
NORTH AMERICA

Embedded Linux Conference

Thank you!