Supply-chain Levels for Software Artifacts (SLSA)

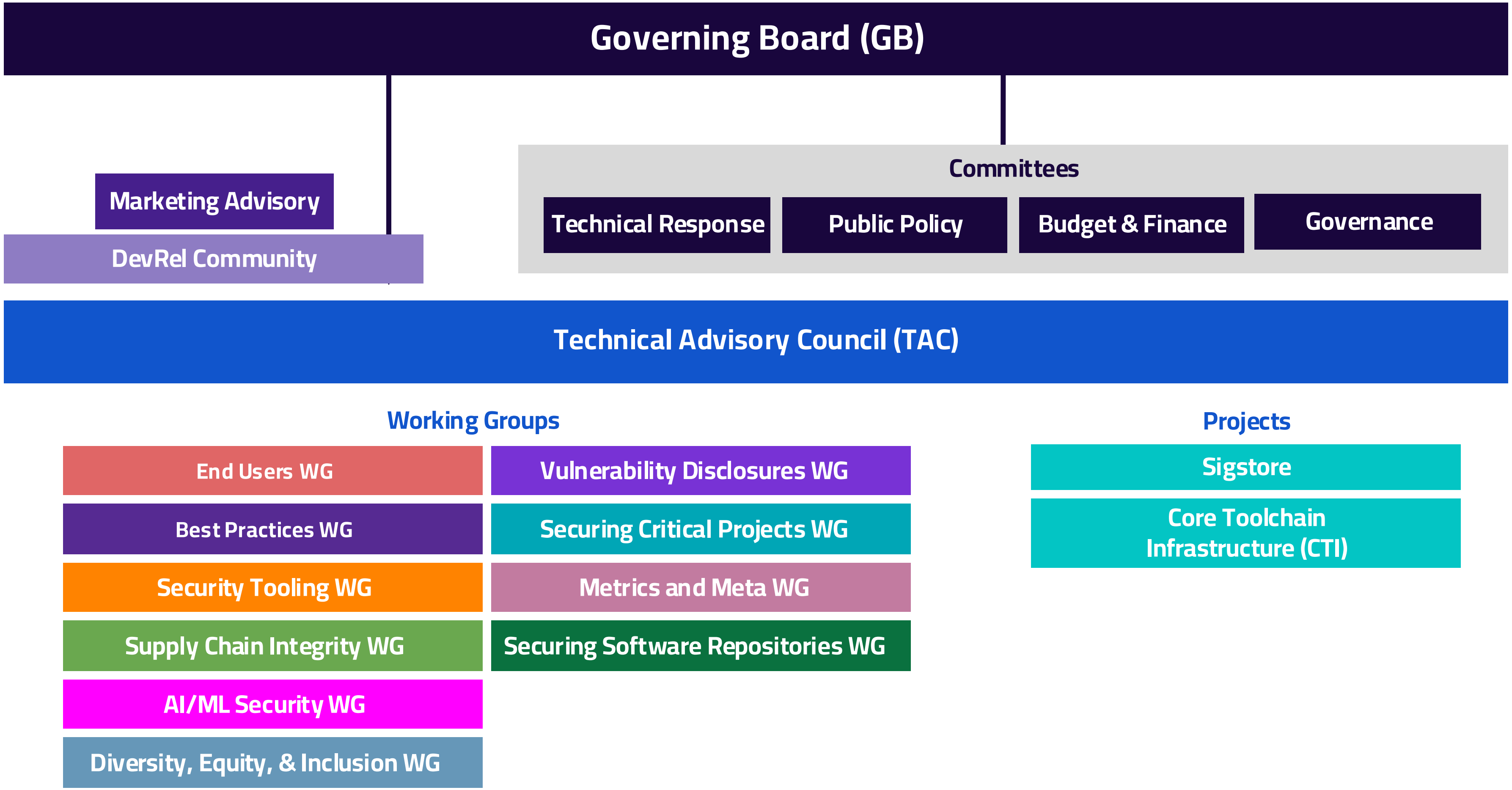
Safeguarding artifact integrity across any software supply chain

Arnaud Le Hors lehors@us.ibm.com

Open Source Security Foundation (OpenSSF)

- OpenSSF is a community of software developers, security engineers, and more who are working together to secure open source software for the greater public good.
- Created in 2020 under the Linux Foundation
- In Jan 2022 switched to member-funded model

OpenSSF Structure



Working Groups, Projects, & SIGs

1. INFORM

Vulnerability Disclosures
Efficient vulnerability reporting and remediation

I. CVD Guides SIGs
J. OSS-SIRT SIG
K. Open Source Vuln Schema (OSV) project
L. OpenVEX project
OpenVEX SIG
M. Vuln Autofix SIG

2. EQUIP

Best Practices
Identification, awareness, and education of security best practices

A. Secure Software Development Fundamentals courses SIG
B. Security Knowledge Framework (SKF) project
C. OpenSSF Best Practices Badge project
D. OpenSSF Scorecard project
E. Common Requirements Enumeration (CRE) project
F. Concise & Best Practices Guides SIGs
G. Education SIG
H. Memory Safety SIG
AG. The Security Toolbelt SIG
AL. Python Hardening SIG

3. ENGAGE

End Users
Voice of public & private sector orgs that primarily consume open source

Z. Threat Modeling SIG

Metrics & Metadata
Security metrics/reviews for open source projects

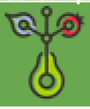
N. Security Insights project
O. Metrics API SIG
P. Security Reviews project

Security Tooling
State of the art security tools

Q. SBOM Everywhere SIG
R. OSS Fuzzing project
AI. SBOMit project
AJ. Protobom project

Supply Chain Integrity
Ensuring the provenance of open source code

S. Supply-chain Levels for Software Artifacts (SLSA) project
T. Secure Supply Chain Consumpt Framework (S2C2F) project
AJ. gittuf project
AK. GUAC project
AM. Zarf project



Securing Software Repositories
collaboration between repository operators

AB. RSTUF Project

Securing Critical Projects
Identification of critical open source projects

U. List of Critical OS Prj, components, & Frameworks SIG
V. criticality_score project
W. Census SIG
X. Package Analysis project
Y. allstar project

AI/ML Security
AI/ML Security at the Intersection of Artificial Intelligence and Cybersecurity

AD. Model Signing SIG

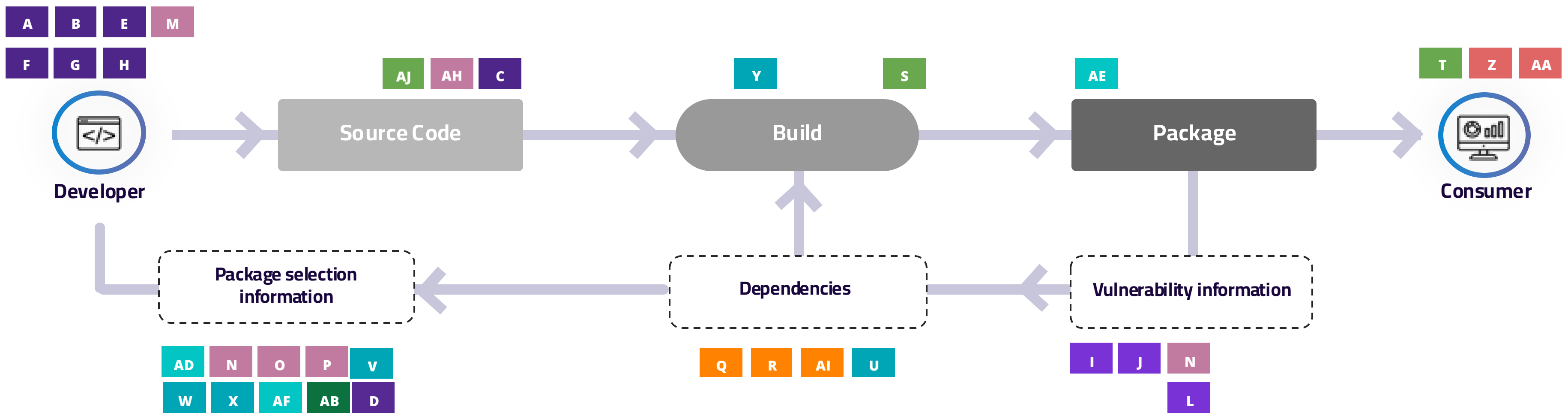
DevRel
Develop Use Cases and help others learn about security

Diversity, Equity, & Inclusion
Increase representation and strengthen the overall effectiveness of the cybersecurity workforce

Projects
Category-leading software initiatives

AE. Sigstore
AF. Core Toolchain Infrastructure (CTI)

How OpenSSF Projects & SIGs Work Together ("CI/CD View")



Best Practices

A. Secure Software Development Fundamentals courses SIG
B. Security Knowledge Framework (SKF) project
C. OpenSSF Best Practices Badge project
D. OpenSSF Scorecard project
E. Common Requirements Enumeration (CRE) project
F. Concise & Best Practices Guides SIGs
G. Education SIG
H. Memory Safety SIG

DevRel Community

Vulnerability Disclosures

I. CVD Guides SIGs
J. OSS-SIRT SIG
K. Open Source Vuln Schema (OSV) project
L. OpenVEX SIG
M. Vuln Autofix SIG

Metrics & Metadata

N. Security Insights
O. Security-Metrics: Risk Dashboard project
P. Security Reviews project
AH. Security Insights Spec project

Security Tooling

Q. SBOM Everywhere SIG
R. OSS Fuzzing SIG
AI. SBOMit project
AJ. Protobom project

Supply Chain Integrity

S. SLSA project
T. S2C2F project
AJ. Gittuf project
AK. GUAC project

Securing Critical Projects

U. List of Critical OS Projects SIG
V. criticality_score project
W. Harvard study SIG
X. Package Analysis project
Y. allstar project

End Users

Z. Threat Modeling SIG

Securing Software Repositories

AB. Repository as a Service Project

AI/ML Security

Diversity, Equity, & Inclusion

Projects

AD. Alpha & Omega project
AE. Sigstore
AF. Core Toolchain Infrastructure (CTI)

Sample OpenSSF Project/SIG Results

- Education: *Secure Software Development Fundamentals* (**free** course)
- Guides:
  - *Concise Guide for Developing More Secure Software*
  - *Concise Guide for Evaluating Open Source Software*
- OSS Security Evaluation:
  - *OpenSSF Scorecard*; auto-measures OSS github.com/ossf/scorecard
  - *OpenSSF Best Practices Badge* (for OSS projects); >6,100 participating, 3 levels
  - *Supply-chain Levels for Software Artifacts (SLSA)*
- Improved tooling: *Sigstore (signing)*
- Vulnerability finding/reporting:
  - *Alpha-Omega:* proactively find/fix vulnerabilities openssf.org/community/alpha-omega
  - *Vulnerability Disclosure Guide* github.com/ossf/oss-vulnerability-guide

SLSA

Supply-chain Levels for Software Artifacts, or SLSA ("salsa").

It’s a security framework, a checklist of standards and controls to prevent tampering, improve integrity, and secure packages and infrastructure. It’s how you get from "safe enough" to being as resilient as possible, at any link in the chain.

If an SBOM is like a list of ingredients, SLSA is like all of the food safety handling guidelines that make an ingredient list credible.

SLSA: Use Cases

SLSA is for everyone involved in producing, consuming, and providing infrastructure for software such as build platforms and package ecosystems.

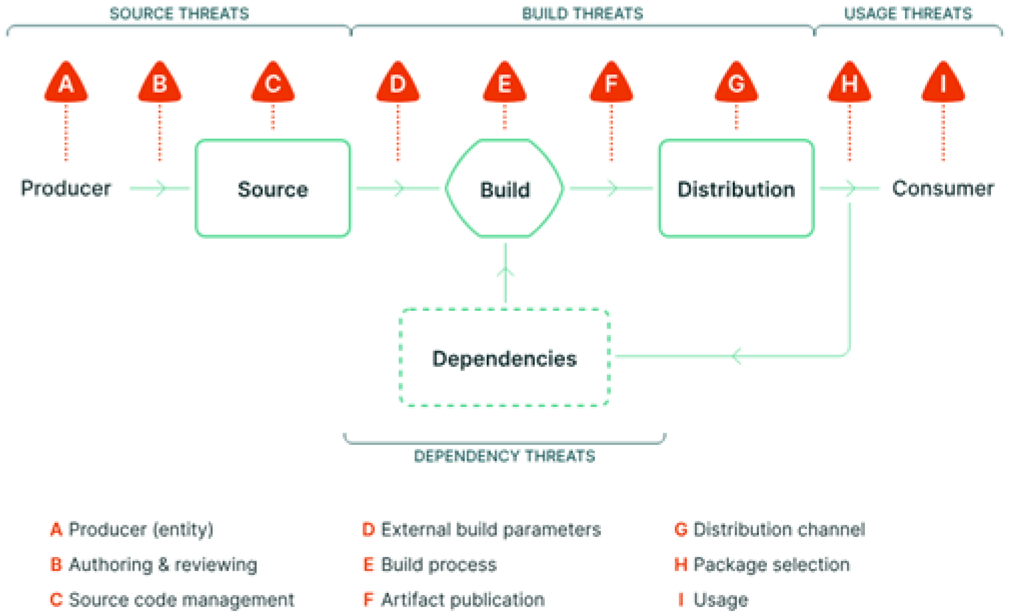
Producers: for protection against tampering and insider threats

Consumers: to verify the software they rely on is secure

Infrastructure Providers: as a guideline for hardening build platforms and processes.

SLSA: The Supply Chain Problem

Any software can introduce vulnerabilities into a supply chain. As a system gets more complex, it's critical to already have checks and best practices in place to guarantee artifact integrity, that the source code you're relying on is the code you're actually using. Without solid foundations and a plan for the system as it grows, it's difficult to focus your efforts against tomorrow's next hack, breach or compromise.



SLSA offers...

- A common vocabulary to talk about software supply chain security
- A way to secure your incoming supply chain by evaluating the trustworthiness of the artifacts you consume
- An actionable checklist to improve your own software's security
- A way to measure your efforts towards compliance with forthcoming standards such as NIST's Secure Software Development Framework (SSDF)

Not another scanner!

- It’s not a security scanner it won’t detect insecure code, CVEs, tokens, passcodes etc.
- It will not
  - Prevent insider threat
  - Prevent typosquatting
  - Reduce introduced insecure code
- It Does
  - Create a document for each build
  - Set common terminology around what security steps are being taken on writing source and performing a build

Quick History

- Started by Google
- Contributed to OpenSSF in 2021
- v0.1 published in June 2021
- v1.0 published in April 2023
- Current version v1.1 published in April 2025

Release Summary V1.0

- Overall, SLSA v1.0 is more stable and better defined than v0.1, but less ambitious.

- It corresponds roughly to the build and provenance requirements of SLSA v0.1 Levels 1 through 3, deferring SLSA Level 4 and the source and common requirements to a future version.

Key Changes

- Uses a stricter language with requirements based on MUST, SHOULD, MAY per RFC 2119
- Division of the *levels* into multiple *tracks* (build, source, deps), which are separate sets of levels that measure different aspects of software supply chain security
- Limited to *Build track* Levels 1-3 for v1.0
- Reorganization and simplification of the core specification to make it easier to understand and to provide better guidance
- New guidance on verification of artifacts and build systems
- Updated Provenance and Verification Summary Attestation (VSA) schemas, with a clearer build model

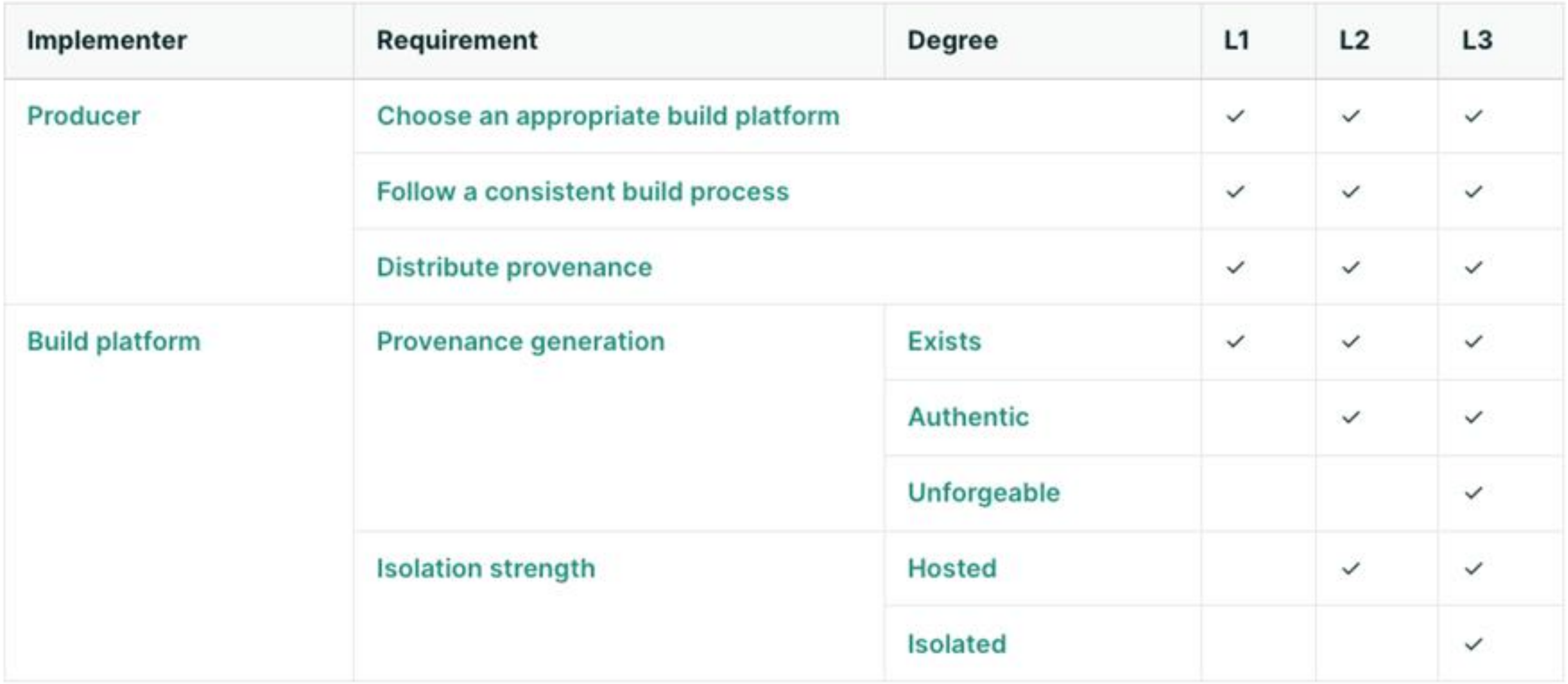
Release Summary V1.1

- Refined threat model
- Adds procedure for verifying Verification Summary Attestations (VSAs)
- Adds verifier metadata to VSA format
- Various minor fixes and clarifications

SLSA Build Track Levels

Level	Requirements	Focus
Build L1	Provenance showing how the package was built	Mistakes, documentation
Build L2	Signed provenance, generated by a hosted build platform	Tampering after the build
Build L3	Hardened build platform	Tampering during the build

SLSA Build Track Levels (cont)



Implementer	Requirement	Degree	L1	L2	L3
Producer	Choose an appropriate build platform		✓	✓	✓
	Follow a consistent build process		✓	✓	✓
	Distribute provenance		✓	✓	✓
Build platform	Provenance generation	Exists	✓	✓	✓
		Authentic		✓	✓
		Unforgeable			✓
	Isolation strength	Hosted		✓	✓
		Isolated			✓

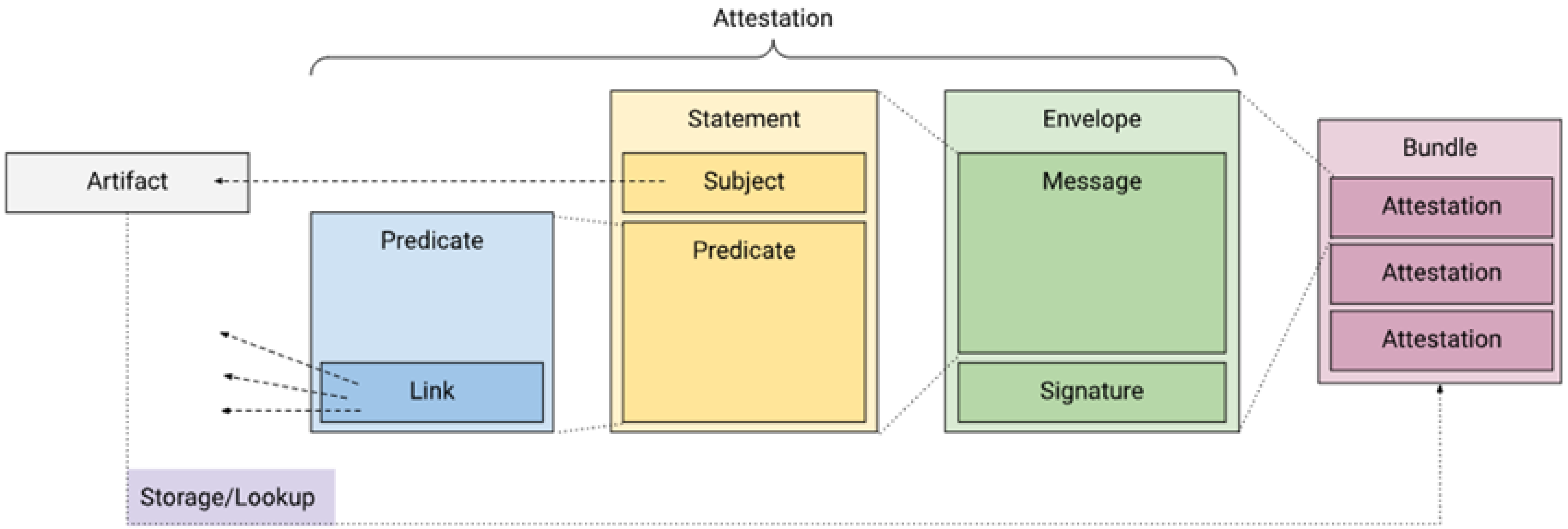
Provenance & Verification Summary Attestations (VSA)

SLSA requires the distribution of provenance metadata along with an artifact to enable verification

- In the form of immutable SLSA attestations bound to artifacts
- Maybe published in various places: source repository, package registry, ...
- Format is open but in-toto SLSA Provenance is recommended

Verification Summary Attestations (VSA)

- Communicates that an artifact has been verified at a specific SLSA level and details about that verification
- Allows consumers to delegate verification to a trusted party



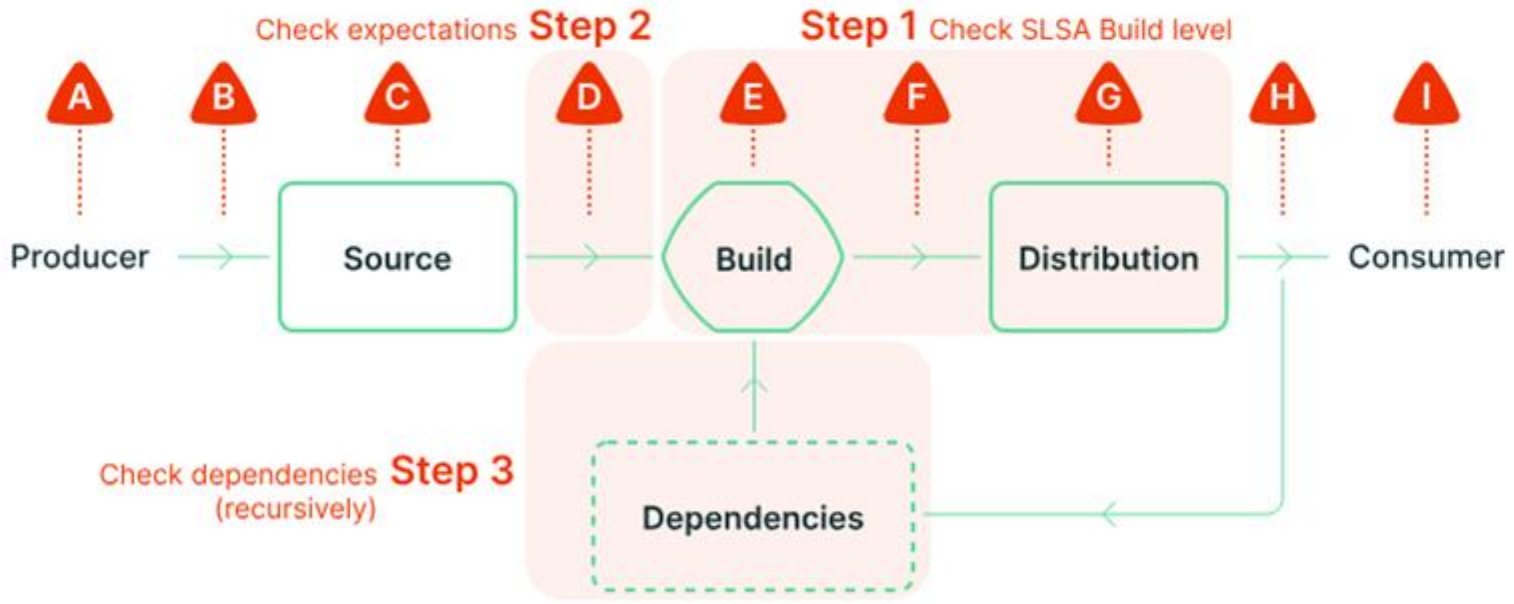
Component	Recommendation
Envelope	DSSE (ECDSA over NIST P-256 (or stronger) and SHA-256.)
Statement	in-toto attestations
Predicate	Choose as appropriate, i.e.; Provenance, SPDX, other predicates defined by third-parties. If none are a good fit, invent a new one
Bundle	JSON Lines, see attestation bundle

Verifying

One of SLSA's guiding principles is to "trust platforms, verify artifacts".

- First establishing trust in build platforms
  - This would be similar to a security questionnaire (re: external parameters, control plane, build environments, caches, and outputs)
  - Once trust is established, you can trust that platform, and verify artifacts from it.
- Verifying artifacts produced by those trusted build platforms.
  - Verifying the signature on the provenance envelope.
  - Check the declared SLSA build level
  - Ensuring that the values match the expected values.

Verifying


Check expectations Step 2
Step 1 Check SLSA Build level
A
B
C
D
E
F
G
H
I
Producer
Source
Build
Distribution
Consumer
Check dependencies Step 3 (recursively)
Dependencies

A Producer (entity)
B Authoring & reviewing
C Source code management
D External build parameters
E Build process
F Artifact publication
G Distribution channel
H Package selection
I Usage

Tools (a few examples)

- Many popular development platforms can generate an Attestation for your code.
    - TestifySec Witness plugin for GitHub and GitLab and more!
    - Google Cloud Build
    - IBM OnePipeline
    - Red Hat Konflux CI
    - GitHub Actions
    - GitLab Runner Attestations
- Want attestations for your OSS?
    - npm now supports attestations for maintainers!
    - ActiveState vendors open source dependencies with SBOM and SLSA attestations

Coming up

- Source Track
  - Describes increasing levels of trustworthiness and completeness in a repository revision’s provenance (e.g. how it was generated, who the contributors were, etc.)
- Build Environment Track
  - Aims to making it possible to validate the integrity and trace the Provenance of core build platform components
- Dependency Track
  - Refactoring of the Secure Supply Chain Consumption Framework (S2C2F)
  - Enables a software producer to easily measure, control and reduce risk arising from third-party dependencies.

See SLSA Working Draft for more info: https://slsa.dev/spec/draft/

Source Track Sneak Peek

- Level 1: Version controlled
  - The source is stored and managed through a modern Version Control System (VCS).
- Level 2: Branch History
  - Clarifies which branches in a repo are consumable and guarantees that all changes to protected branches are recorded.
- Level 3: Authenticatable and Auditable Provenance
  - The Source Control System (SCS) generates credible, tamper-resistant, and contemporaneous evidence of how a specific revision was created.
- Level 4: Two-party review
  - The SCS requires two trusted persons to review all changes to protected branches.

Translates into requirements for the producing Organization and Source Control Systems.

SLSA: Get Involved

Website: https://slsa.dev/

GitHub: https://github.com/slsa-framework/slsa

Slack: #slsa

Working Group: Supply Chain Integrity

This presentation is released under the CC-BY-4.0 license