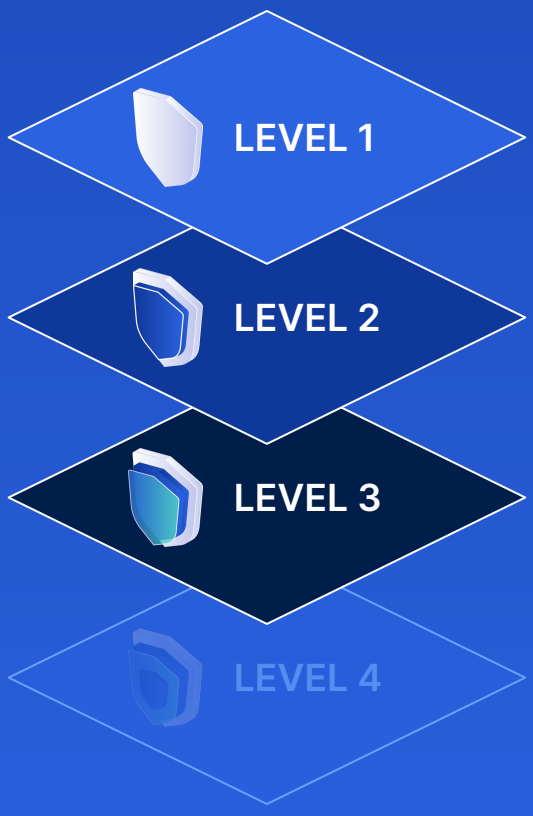
Scribe

A Use Case of Using Scribe Trust Hub

# Reaching SLSA Levels with Scribe's

## `valint slsa`




Easy to adopt, giving you supply chain visibility and being able to generate provenance

Starts to protect against software tampering and adds minimal build integrity guarantees

Hardens the infrastructure against attacks, more trust integrated into complex systems

The highest assurance of build integrity and measures for dependency management in place

# TABLE OF CONTENTS

# BACKGROUND

SLSA (Supply-chain Levels for Software Artifacts) is a security framework aiming to prevent tampering, improve integrity, and secure packages and infrastructure.

The core concept of SLSA is that a software artifact can be trusted only if it complies with three requirements:

The artifact should have a Provenance document describing its origin and building process (L1).

The Provenance document should be trustworthy and verified downstream (L2).

The build system should be trustworthy (L3).

The SLSA framework defines levels, which represent how secure the software supply chain is. These levels correspond to the level of implementation of these requirements (noted as L1-L3 above).

Scribe's `valint slsa` command can be used to produce Provenance documents. Following, we describe how to achieve SLSA levels by using this tool.

Note: We refer here to the SLSA V1.0 framework.

# SLSA L1

The requirements for SLSA L1 include:

- Software producers follow a consistent build process.
- The build platform automatically generates provenance data describing how the artifact was built.
- Software producers distribute provenance data to consumers.

Checklist for achieving SLSA L1:

- ✓ Build your software using a CI system. Preferably use a build script that is source-controlled.
- ✓ Activate the `valint slsa` command as part of your build script to create a Provenance document. Notice that the `valint slsa` command allows adding additional information to the Provenance document - you can tailor the content of the Provenance document to your needs.

# SLSA L2

The requirements for SLSA L2 include:


The SLSA L1 requirements.

The build runs on a hosted build platform that generates and signs the provenance itself.

Downstream verification of provenance includes validating the authenticity of the provenance.

Checklist for achieving SLSA L2:

- ✓ The SLSA L1 checklist.
- ✓ Use a hosted build service (as opposed to performing the build on the developer machine).
- ✓ Create a signed Provenance document (instead of an unsigned one, which is enough for SLSA L1) This can be achieved by running `valint slsa ... -o attest`. Scribe's `valint` tool has a wide set of signing capabilities; for an enterprise, we recommend using X.509 PKI keys and certificates.
- ✓ Verify the authenticity of the Provenance document downstream using the `valint verify` command. Verification may include signature and signing identity, as well as other verification rules that assure the content of the SLSA Provenance document. Some examples can be found in Scribe's policy catalog.

Note: Verification should be done by the consumer of the built software; for internal compliance, the verification can be done as part of a testing pipeline.

# SLSA L3

## SLSA L3 Requirements

The requirements for SLSA L3 include:

L2

The SLSA L2 requirements.

Build platform implements strong controls to:

- Prevent runs from influencing one another, even within the same project.
- Prevent secrets used to sign the provenance from being accessible from the user-defined build steps.

In addition, in order to trust the build platform, one needs to verify the build platform. The build platform should be trusted in the sense that the Provenance document will be unforgeable and the build will be isolated. Such verification derives the following requirements:

Verify that the use of the platform does not break the unforgeability and isolation requirements.

- Verifying the isolation, for example, can be done by evaluating the use of cache in the pipeline.
- To ensure the unforgeability of the Provenance document, we recommend generating and signing it in a dedicated build pipeline.

Verify the trustworthiness of the build platform.

- For SaaS CIs, a verification should be done with the build platform vendor. In cases where the software producer is responsible for the deployment of the build system, a combination of vendor-self-attestation and performing an analysis of the deployment aspects is recommended.
- For example, When deploying a self-hosted CI, the vendor attestation should declare how builds are isolated from each other, and the deployment analysis should verify the access permissions and log-auditing of the CI system.

These requirements are challenging because satisfying them is CI-platform-dependent, cannot be fully automated, and requires a professional security analysis of the build systems and pipelines. That is why the SLSA framework specifically suggests that organizations gradually evolve from SLSA L2 to SLSA L3 compliance.

---

If you made your way through this article all the way up to here and decided that SLSA L3 is the right thing for you, roll up your sleeves - here is our recommendation for a checklist:

## Checklist for achieving SLSA L3:

✅ The SLSA L2 checklist.

✅ Assess the CI system. The goal is to answer the following questions:

- Under what conditions can an unauthorized entity evade the build system?
- Under what conditions can builds affect each other?

Once answered - manage the remaining risks.

✅ Isolate the generation of the Provenance document:

- Separate the creation of the Provenance document to a different pipeline, preferably on a separate build service.
  - Expose to this pipeline only the secrets used for signing the Provenance document.
  - Either create or verify the Provenance document content on this pipeline. In the case of verification, verify all possible fields of an in-pipeline-generated Provenance document with data collected directly from the build platform or from other trusted sources.

✅ Isolate and verify the isolation of the build pipeline from other pipeline runs:

- Verify the use of caches and shared volumes.
- Verify that secrets shared with other pipelines cannot allow for pipelines to affect each other.
- Verify that pipeline runs cannot affect each other
  - For example, prevent installations done through one pipeline from affecting other pipeline runs. This can be done by using ephemeral build-runners (such as a container that is created for each build) or by verifying that build-runners start each time from a predetermined state.

## To achieve SLSA L3 with Scribe tools, we recommend the following:

Instrument the build pipeline for generating all attestations that will be needed to populate the Provenance document. For example, you may decide you want a list of the dependencies installed during the build. This list can be generated by a `valint bom dir:` command. In addition, create a Provenance attestation in the pipeline using the `valint slsa` command.

Create a separate trusted-provenance-generation pipeline that will perform the following:

- Generate a trusted Provenance document based on the one created in the build pipeline;
  - Collect data from the build service and use it to verify and update the Provenance document.
  - Verify the content of attestations created in the build pipeline. For example, verify the content of the build-runner by comparing an SBOM attestation from the build pipeline with an SBOM attestation that was sampled separately.
  - Use attestations collected from the build pipeline to update the Provenance document.
  - Updating the Provenance document can be done using the `valint slsa` command.
- Verify that the build was isolated by evaluating data collected from the build service. For example - verify the use of caches and secrets.

In order to perform such data collection and evaluation, Scribe provides tools that create attestations to the build run and perform the verifications needed.

---

All right. It seems you are all set to go. Of course, if you need any assistance, let us know. We are here to help and would love to advise or actively help.