The ActiveState Approach to Supply chain Levels for Software Artifacts (SLSA)

ActiveState

The ActiveState Approach to Supply chain Levels for Software Artifacts (SLSA)

Open source software's speed and innovation benefits have made it an essential element of modern software development, despite the unique security threats it engenders, such as typosquatting and dependency confusion. These vectors can introduce various forms of malware into an Independent Software Vendor's (ISV) organization, which may then be propagated downstream to its customers.

It's this force multiplier – a single cyberattack on a major ISV that can potentially compromise tens of thousands of end user companies – that caused President Biden to issue his **executive order** aimed at improving the United States' cybersecurity stance. In response, Google launched an internal initiative that has since become an industry-wide collaboration: Supply chain Levels for Software Artifacts or **SLSA**, which is a security framework designed to:

- Prevent tampering within the software development process
- Improve the integrity of built artifacts
- Ensure the security of open source packages
- Secure the infrastructure your projects rely on

ActiveState is committed to helping developers ensure the security and integrity of the open source language packages they use in their software development processes. To that end, we've been creating reproducible builds since 1997, so we're pleased that this essential capability has been identified as a key component in the SLSA framework for securing the software supply chain. But with our ActiveState Platform, we're delivering all the controls required to generate SLSA Level 4 artifacts for the open source language runtime environments your projects run on.

This paper will introduce each SLSA criteria, and detail how ActiveState can help you meet all requirements up to and including the highest level of security and integrity: SLSA Level 4.

BUILD REQUIREMENTS

A build is the process by which source code is assembled into a built software artifact. The build process must be secured against unknown or outside influence to avoid tampering. Builds should always be verifiable and reproducible.

Scripted Builds

Requirements	SLSA1	SLSA2	SLSA3	SLSA4
Build - Scripted build	✓	✓	✓	✓

SLSA Requirement: All build steps must be fully defined in some sort of "build script". The only manual command permitted, if any, invokes the build script.

ActiveState Process: ActiveState uses "builders," which are reusable scripts used to bootstrap native build systems. These are invoked by a build wrapper that records all actions taken by the builders.

Dedicated Build Service

Requirements	SLSA1	SLSA2	SLSA3	SLSA4
Build - Build Service		✓	✓	✓

SLSA Requirement: All build steps run using a dedicated build service, rather than a developer's workstation.

ActiveState Process: ActiveState's cloud-hosted build platform performs all builds from source without developer intervention.

Built as Code

Requirements	SLSA1	SLSA2	SLSA3	SLSA4
Build - Build as code			✓	✓

SLSA Requirement: The build definition and configuration executed by the build service is verifiably derived from text file definitions stored in a version control system.

Verifiably derived can mean either fetched directly through a trusted channel, or that the derived definition has some trustworthy provenance chain linking back to version control.

ActiveState Process: All builds are bootstrapped by ActiveState-authored build scripts, which are hosted in a private git repository.

Ephemeral Environments

Requirements	SLSA1	SLSA2	SLSA3	SLSA4
Build - Ephemeral Environments			✓	✓

SLSA Requirement: The build service ensures that the build steps run in an ephemeral environment, such as a container or VM provisioned solely for this build, are not reused from a prior build.

ActiveState Process: ActiveState uses a mix of ephemeral containers (Windows, Linux) and VMs (macOS) for all builds. All containers are discarded at the completion of each build step.

Isolated Builds

Requirements	SLSA1	SLSA2	SLSA3	SLSA4
Build - Isolated			✓	✓

SLSA Requirement: The build service ensures that the build steps run in an isolated environment free of influence from other build instances, whether prior or concurrent.

ActiveState Process:

- ActiveState's build containers (Windows, Linux) and VMs (macOS) are isolated from one another, and cannot contaminate other builds.
- ActiveState follows industry best practices for handling signing keys. Signing keys are not available during artifact builds.
- ActiveState's container and VM isolation prevents builds from affecting one another either concurrently or subsequently.
- ActiveState's artifacts are cached in a cloud-hosted, content-addressable storage solution.

Parameterless Builds

Requirements	SLSA1	SLSA2	SLSA3	SLSA4
Build - Parameterless				✓

SLSA Requirement: The build output cannot be affected by user parameters other than the build entry point and the top-level source location. In other words, the build is fully defined through the build script and nothing else.

ActiveState Process: ActiveState Platform users have no direct control over build invocation or the build environment. The Platform manages all required build resources and invocations automatically.

Hermetic Builds

Requirements	SLSA1	SLSA2	SLSA3	SLSA4
Build - Hermetic				✓

SLSA Requirement: All transitive build steps, sources, and dependencies are fully declared up front with immutable references, and the build steps run with no network access.

ActiveState Process:

- The ActiveState Platform's dependency solver and build orchestrator ensure all dependencies are resolved and satisfied prior to build invocation. Build scripts cannot modify any such references.
- The ActiveState Platform's Inventory system maintains all metadata and relationships in perpetuity, and maintains a history for all metadata.
- ActiveState's build artifacts are stored in and served from secure cloud storage.
- All artifacts, including source code are fetched using secure channels from secure cloud storage.
- The build wrapper verifies the hash of all dependent artifacts, and validates the integrity of the build system itself prior to invoking the build script(s).
- ActiveState builds are executed in a heavily protected cloud-hosted build environment.

Reproducible Builds

Requirements	SLSA1	SLSA2	SLSA3	SLSA4
Build - Reproducible				O

SLSA Requirement: Re-running the build steps with identical input artifacts results in bit-for-bit identical output. Builds that cannot meet this criteria MUST provide a justification why the build cannot be made reproducible (currently this requirement is designated as "best effort").

ActiveState Process: ActiveState artifacts generated from the same inputs will contain the same content, although the checksums of the artifacts themselves may differ due to metadata stored in the archive formats used. In general, however, if any build is requested of the platform with identical inputs, the platform will detect this and serve the appropriate cached artifact, saving build time.

PROVENANCE REQUIREMENTS

In the software industry, provenance refers to the traceability of components (e.g., ability to trace and verify the source/origin of a component), as well as built artifacts produced during the development process (e.g., ability to trace and verify the build process). Here, provenance refers to the ability to prove a build has been completed according to the requirements of the SLSA framework, which is established by providing metadata that allows for this to be verified.

Available

Requirements	SLSA1	SLSA2	SLSA3	SLSA4
Build - Available	✓	✓	✓	✓

SLSA Requirement: The provenance is available to the consumer in a format that the consumer accepts. The format SHOULD be in-toto SLSA Provenance, but another format MAY be used if both producer and consumer agree and it meets all the other requirements.

ActiveState Process: Provenance information is available for build artifacts and source code from ActiveState via several of its publicly available APIs. ActiveState will provide this information from a single source in SLSA's proposed in-toto ITE-6 attestation format.

Authenticated

Requirements	SLSA1	SLSA2	SLSA3	SLSA4
Build - Authenticated		✓	✓	✓

SLSA Requirement: The provenance's authenticity and integrity can be verified by the consumer. This SHOULD be through a digital signature from a private key accessible only to the service generating the provenance.

ActiveState Process: ActiveState maintains a secure code signing system, which will be used to sign the attestations.

Service Generated

Requirements	SLSA1	SLSA2	SLSA3	SLSA4
Build - Service Generated		✓	✓	✓

SLSA Requirement: The data in the provenance MUST be obtained from the build service (either because the generator is the build service, or because the provenance generator reads the data directly from the build service).

ActiveState Process: All provenance information is generated automatically by the ActiveState Platform from source code ingestion to artifact generation.

Non-Falsifiable

Requirements	SLSA1	SLSA2	SLSA3	SLSA4
Build - Non Falsifiable			✓	✓

SLSA Requirement: Provenance cannot be falsified by the build service's users.

ActiveState Process:

- End users have no way to influence the provenance data created by the ActiveState Platform. Digital signatures on the provided attestations will guarantee that any tampering with this data is immediately apparent.
- ActiveState maintains a secure code signing system which will be used to sign the above mentioned attestations.
- ActiveState's secure code signing system is completely isolated from the remainder of the platform.
- All build steps and provenance generation are performed within ActiveState's secure, cloud-hosted infrastructure.
- While the output artifact's contents are difficult to predict prior to building, no user-supplied code is used in generating the output artifact's hash. This is calculated separately by the ActiveState Platform once the build script has completed.
- The ActiveState Platform is predicated on never allowing non-reproducible builds.

Dependencies Complete

Requirements	SLSA1	SLSA2	SLSA3	SLSA4
Build - Dependencies Complete				✓

SLSA Requirement: Provenance records all build dependencies that were available while running the build steps. This includes the initial state of the machine, VM, or container of the build worker.

ActiveState Process: All of this information is recorded by the ActiveState Platform and made available via several publicly available APIs. It will also be available in our in-toto ITE-6 attestation format.

REQUIREMENTS ON THE CONTENTS OF THE PROVENANCE

Solutions must be able to show the provenance or original source for an artifact/metadata used in the build process.

Identifies Artifact

Requirements	SLSA1	SLSA2	SLSA3	SLSA4
Identifies Artifact	✓	✓	✓	✓

SLSA Requirement: The provenance MUST identify the output artifact via at least one cryptographic hash.

ActiveState Process: All output artifacts are hashed with SHA-256 automatically by the ActiveState Platform on generation.

Identifies Builder

Requirements	SLSA1	SLSA2	SLSA3	SLSA4
Identifies Builder	✓	✓	✓	✓

SLSA Requirement: The provenance identifies the entity that performed the build and generated the provenance.

ActiveState Process: All of this information is recorded by the ActiveState Platform and made available via several publicly available APIs. It will also be available in our in-toto ITE-6 attestation format.

Identifies build instructions

Requirements	SLSA1	SLSA2	SLSA3	SLSA4
Identifies build instructions	✓	✓	✓	✓

SLSA Requirement: The provenance identifies the top-level instructions used to execute the build.

ActiveState Process: The ActiveState Platform uses scripts called “builders” to create artifacts. The specific builder, entry point, and parameters used to execute the build are recorded and made available via our public API. It will also be available in our in-toto ITE-6 attestations.

Identifies Source Code

Requirements	SLSA1	SLSA2	SLSA3	SLSA4
Identifies Source Code		✓	✓	✓

SLSA Requirement: The provenance identifies the repository origin(s) for the source code used in the build.

ActiveState Process: All of this information is recorded by the ActiveState Platform and made available via several publicly available APIs. It will also be available in our in-toto ITE-6 attestation format.

Identifies Entry Point

Requirements	SLSA1	SLSA2	SLSA3	SLSA4
Identifies Entry Point			✓	✓

SLSA Requirement: The provenance identifies the “entry point” of the build definition (see build-as-code defined above) used to drive the build, including which source repo the configuration was read from.

ActiveState Process: All of this information is recorded by the ActiveState Platform and made available via several publicly available APIs. It will also be available in our in-toto ITE-6 attestation format

Includes All Build Parameters

Requirements	SLSA1	SLSA2	SLSA3	SLSA4
Includes All Build Parameters			✓	✓

SLSA Requirement: The provenance includes all build parameters under a user’s control.

ActiveState Process: All of this information is recorded by the ActiveState Platform and made available via several publicly available APIs. It will also be available in our in-toto ITE-6 attestation format.

Includes All Transitive Dependencies

Requirements	SLSA1	SLSA2	SLSA3	SLSA4
Includes all transitive dependencies				✓

SLSA Requirement: The provenance includes all transitive dependencies listed in “Dependencies Complete.”

ActiveState Process: The ActiveState Platform records all direct dependencies of an artifact as references to those artifacts. Each of these dependent artifacts will have its own attestation which can be requested via the aforementioned references. Hence, it will be possible to walk the entire graph of dependencies for an artifact from its attestation.

Includes Reproducible Info

Requirements	SLSA1	SLSA2	SLSA3	SLSA4
Includes reproducible info				✓

SLSA Requirement: The provenance includes a boolean indicating whether the build is intended to be reproducible and, if so, all information necessary to reproduce the build.

ActiveState Process: The ActiveState platform is designed to actively disallow unreproducible builds, so all artifact provenance will be indicated as reproducible.

Includes Metadata

Requirements	SLSA1	SLSA2	SLSA3	SLSA4
Includes metadata	O	O	O	O

SLSA Requirement: The provenance includes metadata to aid debugging and investigations. This SHOULD at least include start and end timestamps, and a unique identifier to allow finding detailed debug logs (currently this requirement is marked as “o” which means “best effort”).

ActiveState Process: The ActiveState platform records and stores all of the specified data for each commit, although it has not yet surfaced in our user interface.

Where are we headed next?

ActiveState is actively participating in the SLSA working groups and keeping a close eye on the industry **standardization effort** being driven by several industry stakeholders. While the standard is still in its infancy, ActiveState is closely watching how it evolves in order to align our current practices with the emerging framework.

Software supply chain security is key to eliminating vulnerabilities and vectors of attack that threaten the security and integrity of software applications. Supply chain attacks are not new, but as open source usage has grown, it's more important than ever to ensure you are importing uncompromised code, building artifacts securely and distributing uninfected software to your customers. In the face of incidents like **SolarWinds, Codecov**, and **many others**, it's time to reestablish trust in the software supply chain, and implementing SLSA is a key first step.

Learn more about how ActiveState can help secure your Open Source Supply Chain
https://www.activestate.com/solutions/slsa

Book a demo and let us show you how the ActiveState Platform can secure your Software Supply Chain:
https://www.activestate.com/solutions/enterprise-security/

Talk to us

ActiveState®

www.activestate.com

Toll-free in NA: 1-866.631.4581

solutions@activestate.com