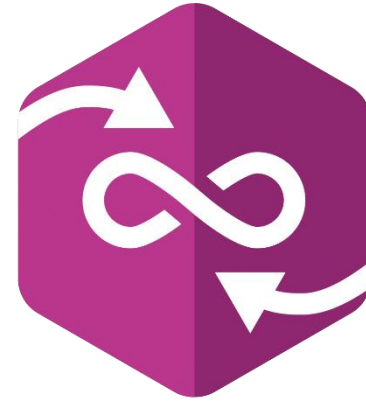


Supply Chain Security using SLSA

GDS Central | 7 Aug 2023



shiphats

Agenda

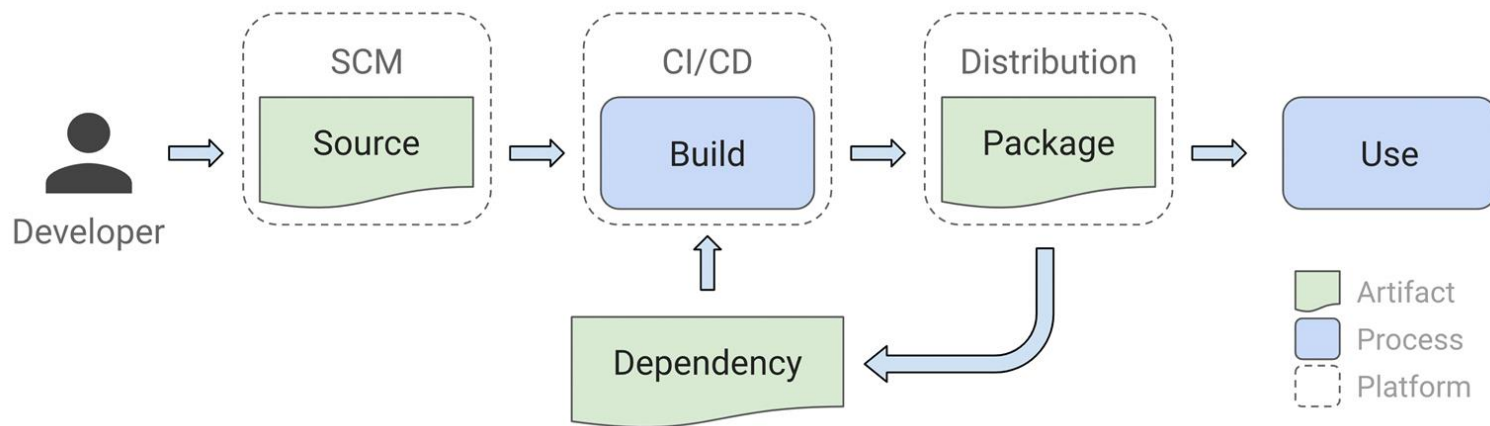
- Introduction to Software Supply Chain
- Software Bill of Materials (SBOM)
- SLSA Security Levels & Requirements
- SLSA in SHIP-HATS + Demo

Introduction to Software Supply Chain

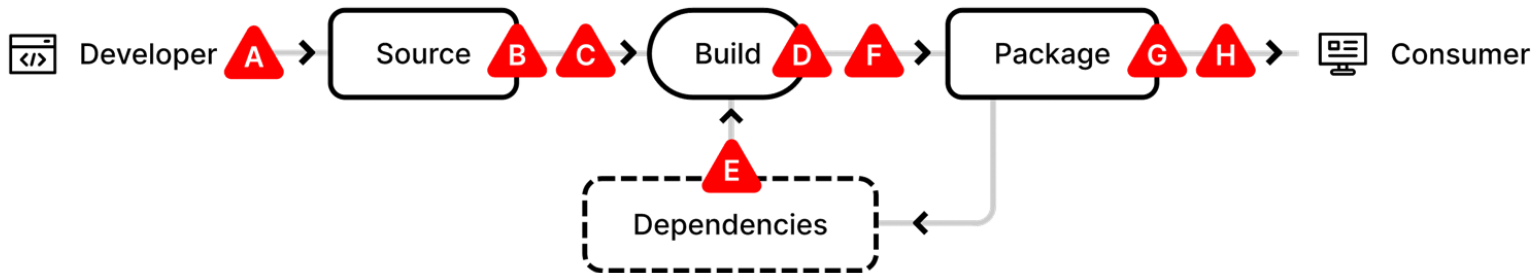


Introduction to Software Supply Chain

- A **software supply chain** is composed of the components, libraries, tools, and processes used to develop, build, and publish a software artifact
- The sequence of steps resulting in the creation of an artifact



Supply Chain risks at the Source, Build and Deploy



SOURCE THREATS

- A Bypassed code review
- B Compromised source control system

BUILD THREATS

- C Modified code after source control
- D Compromised build platform
- F Bypassed CI/CD
- G Compromised package repo
- H Using a bad package

DEPENDENCY THREATS

- E Using a bad dependency

Software Attestation

- **Software Attestation** is a signed document that associates metadata with an artifact. It aims to provide trusted context for software artifacts.
- In SLSA, software attestations are generated during the build process and usually includes details such as:
 - The origins of the source code – its git repository, branch, tag, etc.
 - The build process from which it was created – the CI system, pipeline/job, etc.
 - The external dependencies used to build it (SBOM)
 - The security checks that it passed.
- GitLab supports automated attestation generation in Release 15.1

Software Bill of Materials (SBOM)



Software Bill of Material (SBOM)

- **SBOM** is a nested inventory, a list of ingredients that make up software components
- Value for org:
 - Contextualise SBOM data with additional data elements to inform the risk posture of the entity.
 - Improve visibility and integrating vulnerability detection with SBOM for quicker response to risk in the supply chain



SBOM formats

	CycloneDX	SPDX
Organization	A “meritocratic, consensus-based community project” with a Industry Working Group	SPDX Workgroup (~20 orgs) under the Linux Foundation
Initial draft	2017	2010
Format	XML, JSON	RDF, XLS, SPDX, YAML, JSON
Spec	github.com/CycloneDX/specification	spdx.github.io/spdx-spec BS ISO/IEC 5962 – 2020
Original use cases	For use with OWASP Dependency-Track	License management
Unique features	Extensible format and integrates SPDX license IDs, pURL, and other external identifiers	Extensive support for expressing license details

SLSA Security Levels & Requirements



What is SLSA

- **Supply chain levels for software artifacts (SLSA)** is a set of incrementally adoptable guidelines for supply chain security, established by industry consensus.
- SLSA is organized into a series of levels that provide increasing supply chain security guarantees. This gives you confidence that software hasn't been tampered with and can be securely traced back to its source.
- SLSA v1.0 defines the SLSA Build track to begin this separation of requirements, with other tracks to come in future versions.

SLSA Levels and Tracks

- Three tracks:
 - **Build track** describes increasing levels of trustworthiness and completeness in a package artifact's provenance.
 - **Source track** could provide protection against tampering of the source code prior to the build.
 - **Build Platform Operations track** could provide assurances around the hardening of build platforms as they are operated.
- SLSA levels are split into tracks

SLSA Build Track Levels

Track/Level	Requirements	Focus
Build L0	No guarantees.	(n/a)
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 Quick Wins using SHIP-HATS GitLab

Requirement	Degree	L1	L2	L3	
Choose an appropriate build platform		✓	✓	✓	} Leverage Gitlab CI/CD
Follow a consistent build process		✓	✓	✓	
Distribute provenance		✓	✓	✓	
Provenance generation	Exists	✓	✓	✓	} Leverage Gitlab Attestations
	Authentic		✓	✓	
	Unforgeable			✓	
Isolation Strength	Hosted		✓	✓	} Leverage Gitlab CI/CD
	Isolated			✓	

Build Level 1 achieved

SLSA Quick Wins using SHIP-HATS GitLab

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

Leverage Gitlab CI/CD

Leverage Gitlab Attestations

Leverage Sigstore, a set of open-source tools to sign/verify software (including the Gitlab attestations)

Leverage Gitlab CI/CD

Build Level 2 achieved

SLSA Quick Wins using SHIP-HATS GitLab

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

Level 3 unattainable for now

Gitlab has to make changes to their platform to ensure it is Level 3 compliant.

SLSA (with SBOM) is not a silver bullet

- SLSA does not prevent:
 - Zero-day attack (e.g., Log4j vulnerability), but with **SBOM** can identify affected artefact faster when made known, and quickens response
- COTS/SaaS requires **SBOM** from vendors for attestation

SLSA in SHIP-HATS + Demo



SLSA in SHIP-HATS

- Focus on the CI/CD pipelines, build process
- Use SHIP-HATS templates
- Grouped into 2 portions:



when building artifacts



when deploying artifacts

****Signing and verifying helps to ensure the integrity of the build process and the deployed artifacts**

SLSA in SHIP-HATS



Items: Artifact, Gitlab Attestation, SBOM

1. Building

- Build artifact
- Generate Gitlab Attestation
- Generate SBOM

2. Signing

- Sign artifact
- Sign attestation
- Sign SBOM

3. Verifying

- Verify artifact
- Verify attestation
- Verify SBOM (optional)

4. Deploying

- Deploy verified artifact

SLSA in SHIP-HATS

	Signing	Verification
If your build artifact is a container image	Template: sign-container https://sgts.gitlab-dedicated.com/wog/gvt/ship/ship-hats-templates/-/tree/main/templates#template-sign-container	Template: verify-container https://sgts.gitlab-dedicated.com/wog/gvt/ship/ship-hats-templates/-/tree/main/templates#template-verify-container
If your build artifact is a file	Template: sign-artifact https://sgts.gitlab-dedicated.com/wog/gvt/ship/ship-hats-templates/-/tree/main/templates#template-sign-artifact	Template: verify-artifact https://sgts.gitlab-dedicated.com/wog/gvt/ship/ship-hats-templates/-/tree/main/templates#template-verify-artifact

Important note: The urls above are for the *main* branch of the SHIP-HATS templates repository, which may be unstable/in development. Please use templates from the latest release tag (e.g **v1.0.8**) of the repository.

SLSA in SHIP-HATS

- Sample pipeline for container artifact
- Pipeline: <https://sgts.gitlab-dedicated.com/wog/gvt/hats/slsa/sample-be-app/-/pipelines/1362250>
- Gitlab CI file: <https://sgts.gitlab-dedicated.com/wog/gvt/hats/slsa/sample-be-app/-/blob/main/.gitlab-ci/template-pipeline.yml>

SLSA in SHIP-HATS

Steps in the sample pipeline

Step	Job name	Description
1	template-build-image	– Build the docker container image and upload it to the Gitlab Container Registry – Generate a Gitlab attestation for the docker image
2	template-nexus-iq-scan	– Example of using Nexus IQ to scan the source code, and generate a Software Bill of Materials (SBOM)
3	template-trivy-scan	– Example of using Trivy to scan the docker image (in the container registry), and generate a Software Bill of Materials (SBOM)
4	template-merge-sbom	– Example of using CycloneDX CLI to merge the 2 SBOMs to form a single SBOM

SLSA in SHIP-HATS

Steps in the sample pipeline

Step	Job name	Description
5	template-sign-image	- Generate Cosign key pair to use for signing and verifying - Sign the container image, generating a signature, and saving the signature in the Gitlab Container Registry - Sign the Gitlab attestation, link it to the container image, and save the signed Gitlab attestation in the Gitlab Container Registry - Sign the merged SBOM, link it to the container image, and save the signed SBOM in the Gitlab Container Registry
6	template-verify-image	- Verify the container image is valid - Verify the signed attestation is valid and belongs to the container image - Verify the signed SBOM is valid and belongs to the container image
7	template-deploy-image	- Just a fake deploy job

SLSA in SHIP-HATS

- Sample pipeline for file artifact
- Pipeline: <https://sgts.gitlab-dedicated.com/wog/gvt/hats/slsa/sample-fe-app/-/pipelines/1365839>
- Gitlab CI file: <https://sgts.gitlab-dedicated.com/wog/gvt/hats/slsa/sample-fe-app/-/blob/main/.gitlab-ci/template-pipeline.yml>

SLSA in SHIP-HATS

Steps in the sample pipeline

Step	Job name	Description
1	template-build-package	- NPM build the static files and zip them up - Generate a Gitlab attestation for the zipped file
2	template-nexus-iq-scan	- Example of using Nexus IQ to scan the source code, and generate a Software Bill of Materials (SBOM)
3	template-sign-package	- Generate Cosign key pair to use for signing and verifying - Sign the zipped file, generating a signature locally - Sign the Gitlab attestation, generating a signed file locally - Sign the NexusIQ SBOM, generating a signed file locally
4	template-verify-package	- Verify the zipped file is valid - Verify the signed attestation is valid and belongs to the zipped file - Verify the signed SBOM is valid and belongs to the zipped file
5	template-deploy-package	- Just a fake deploy job

SLSA in SHIP-HATS

- SLSA has been integrated with Compliance Framework v1.0.3 in June, looking for pilot projects!

Security Feature: Software Levels for Security Artifacts (SLSA)

SLSA, Software Bill of Materials (SBOM), and robust attestation mechanisms have been integrated with the latest Compliance Framework v1.0.3 and applied to E2E templates. With the Compliance Framework v1.0.3 and E2E templates, the SBOM signing and verification will be automatically included in your pipeline. However, if you are using only Compliance Framework, you must generate the attestation in your build job and pass the required variables.

[\[Learn more\]](#)

- E2E Templates: <https://sgts.gitlab-dedicated.com/wog/gvt/ship/e2e-templates>
- Compliance Framework: <https://sgts.gitlab-dedicated.com/wog/ship-hats-compliance>
- More Info about SLSA: <https://docs.developer.tech.gov.sg/docs/ship-hats-docs/getting-started/slsa>

More to Come...

- Verify contents of attestation using CUE or REGO policy
 - <https://docs.sigstore.dev/cosign/attestation/#validate-in-toto-attestations>
- Saving signed metadata/signatures into immutable transparency log server (Rekor)
 - <https://docs.sigstore.dev/rekor/overview/>
- Displaying SLSA scorecard in SHIP-HATS DevSecOps Maturity Report
 - <https://docs.developer.tech.gov.sg/docs/ship-hats-docs/getting-started/metrics>

Have more
questions?

- [Supply Chain Levels for Software Artifacts \(SLSA\)](#)
- Reach out to us at <https://go.gov.sg/she>
- Technical documentation <https://go.gov.sg/ship-hats-docs>

Thank You

