xygeni.

Artifacts are opaque blobs. Their provenance is not.

Prove every artifact came from your source and your pipeline, untampered.

Xygeni Build Security generates signed in-toto and SLSA attestations in your pipelines, stores them with a transparency log, and verifies every artifact before it is installed or run.

SLSA provenance and in-toto attestations, signed keyless or with your own keys, verified before deploy.

xygeni.

THE PROBLEM

You cannot inspect an artifact. You can prove how it was made.

A build artifact is an opaque blob: scanning it says little about whether it was built from the expected source, by the expected pipeline, without substitution on the way to production. SolarWinds-class attacks live exactly there. Customers and regulators now ask for provenance: SLSA, the EU Cyber Resilience Act and US Executive Order 14028.

Xygeni Build Security records who built what, when, how and from which source as signed attestations, keeps them in a registry with a transparency log, and verifies them downstream, including attestations from the signing tools you already use.

FIVE QUESTIONS MOST TEAMS CANNOT ANSWER TODAY

Source
Was this artifact built from the commit we reviewed?

Pipeline
Which pipeline, and which steps, produced it?

Substitution
Was it replaced between the registry and the deploy?

Signing
Who signed it, and can we trust that signer?

Evidence
Can we show SLSA provenance to a customer?

1 command
for one-shot SLSA provenance in any CI pipeline.

Keyless
signing with ephemeral keys and certificates, or key-based with your own keys.

Interop
verifies cosign, witness and GitHub artifact attestations, and Sigstore bundles.

ONE PRODUCT, THREE STEPS

Attest. Store. Verify.

ATTEST
SALT in CI
Signed in-toto and SLSA attestations per pipeline step, linking materials to artifacts.

→

STORE
Registry + log
Archivista storage and a Sigstore Rekor transparency log, with results reported in Xygeni.

→

VERIFY
Before deploy
salt verify checks the artifact against its attestation and policy, including third-party attestations.

Attest every build. Store the evidence. Verify before it runs.

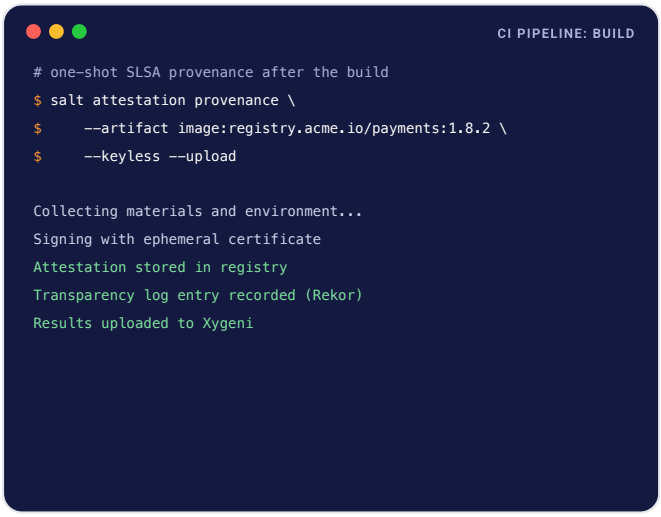
Without Xygeni	With Xygeni
Signing is ad hoc, team by team	**Signed attestations in every pipeline**
No link between source and artifact	**Materials and artifacts linked per build step**
Substitution found after the incident	**Artifact verified against its attestation before deploy**
Provenance requested, nothing to show	**SLSA provenance and attestation policies**

ATTEST · SALT

Record how every artifact was made, as signed evidence

SALT generates in-toto and SLSA attestations in your pipelines, in one shot or step by step.

SALT (Software Attestation Layer for Trust) runs in your CI/CD pipelines and records the environment, the process, the materials that went in and the artifacts that came out of each step. Each attestation is a signed DSSE envelope in the in-toto Attestation Framework, keyless or with your own keys.



Schematic: one-shot SLSA provenance with keyless signing, stored and reported in one step.



Schematic: what an attestation records for a build step.

- **One-shot provenance:** a single command generates SLSA provenance for a build.
- **Keyless signing:** ephemeral key pair and certificate, nothing to manage.
- **Predicates and reports:** SLSA Provenance and other in-toto predicate types, with SBOMs and other reports attached.
- **Incremental attestation:** init, add, run and commit across pipeline steps.
- **Key-based signing:** generate and use your own keys.
- **Source recorded:** repository, ref or tag and build configuration in every attestation.

SLSA Provenance · in-toto · DSSE · GitHub Actions · GitLab CI · Jenkins · Linux, macOS, Windows

Start small, then enforce. One-shot provenance is a single CLI step on your key pipelines. Incremental attestation and policies come next, when you are ready.

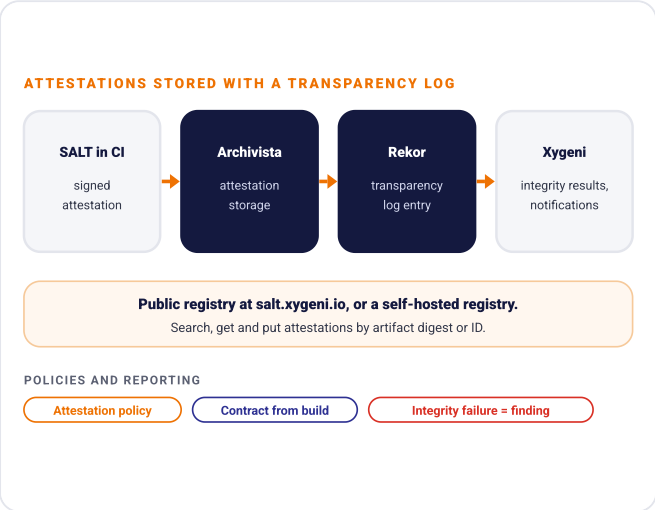
For the platform team: provenance added to the pipeline in one step, not a signing project.

STORE · REGISTRY

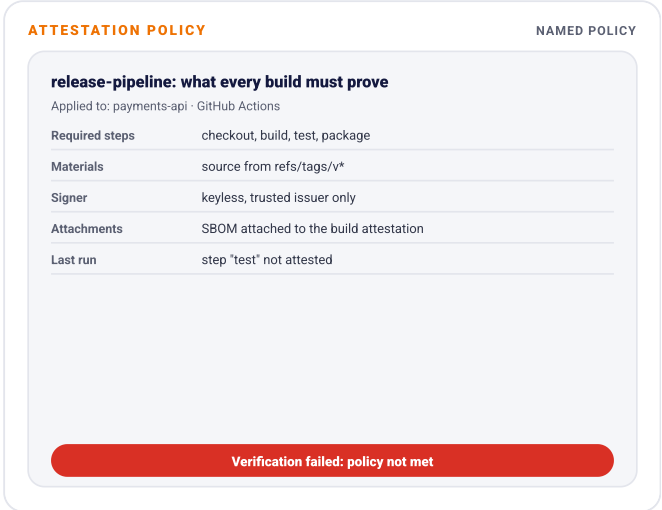
Keep every attestation in a registry backed by a transparency log

in-toto Archivista for storage, Sigstore Rekor for the transparency log, policies for what must be attested.

Attestations are stored in an attestation registry: in-toto Archivista for storage and Sigstore Rekor as a transparency log. Use the public server at salt.xygeni.io or a self-hosted registry. Results are uploaded to Xygeni, so an integrity failure or an error in the attestation process is reported and notified like any other finding.



Schematic: attestations flow from SALT to the registry and the transparency log, and results to Xygeni.



Schematic: a named attestation policy, and a build that does not meet it.

- **Attestation registry:** search, get and put attestations by artifact digest or ID.
- **Public or self-hosted:** salt.xygeni.io or your own registry.
- **Contracts:** create one from a build, then upload and download it.
- **Transparency log:** signed attestations recorded in Sigstore Rekor.
- **Attestation policies:** named policies define what must be attested and the pipeline layout.
- **Reported in Xygeni:** integrity failures and errors raised as findings and notifications.

in-toto Archivista · Sigstore Rekor · salt.xygeni.io · Self-hosted registry

Policies turn attestation into a control. A policy states which steps must be attested and under which conditions, so a build that skips a step fails verification instead of passing unnoticed.

For the CISO: provenance evidence a customer or a regulator can check.

VERIFY · BEFORE DEPLOY

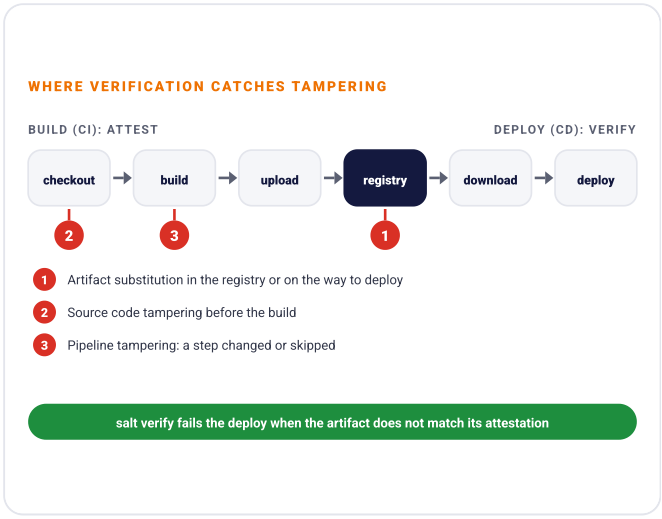
Check every artifact before it is installed or run

salt verify, with certificate and transparency-log checks, interoperable with the signing tools you already use.

Downstream, salt verify checks an artifact against its attestation and its policy before deploy. Keyless attestations are verified through the embedded certificate, signer certificates are validated, untrusted keys are rejected and transparency-log entries are checked. It also verifies attestations from cosign, witness and GitHub artifact attestations, and reads Sigstore bundles as published.



Schematic: salt verify in a deploy pipeline stops an artifact that does not match its attestation.



Schematic: where verification catches artifact substitution, source tampering and pipeline tampering.

- **Verify before use:** the artifact checked against its attestation and policy in the deploy pipeline.
- **Certificate validation:** signer certificates validated, untrusted keys rejected.
- **Interoperable:** cosign, witness, GitHub artifact attestations and Sigstore bundles.
- **Keyless verification:** through the embedded certificate, with no public key to distribute.
- **Transparency-log checks:** SCT presence and automatic Rekor entry lookup.
- **Deterministic by design:** an artifact either matches its attestation or it does not.

CI/CD Security
Hardens the pipeline and SCM configuration that produces the artifact.

Anomaly Detection
Detects tampering with build files and suspicious pipeline activity.

SCA
SBOMs that can travel attached to the attestation of the build that produced them.

For the release manager: a deploy step that asks for proof before it ships.

xygeni.

WHY XYGENI FOR BUILD SECURITY

Attest every build. Store the evidence. Verify before it runs.

Xygeni Build Security gives every artifact verifiable provenance, from the source and the pipeline that produced it to the deploy that uses it, and fits into the signing ecosystem you already run.

Full lifecycle

Generate, store with a transparency log and verify, not just sign.

Works with your stack

Verifies cosign, witness, GitHub and Sigstore attestations.

Policies, not just signatures

Attestation policies define what every build must prove.

In the platform

Integrity results and failures next to the rest of your supply-chain posture.

Why security leaders choose Xygeni

Proven in production

Banco Serfinanza reports around 90% less time identifying, prioritizing and assigning findings; Fintonic, a 90% reduction in prioritization time.

One platform

Code, open source, secrets, IaC, pipelines, running apps and AI in one inventory, one risk model and one workflow.

Extend, don't replace

Xygeni's AI layer triages, explains and fixes findings from the security tools you already run.

Built for regulated environments

European vendor. SaaS, on-premise or air-gapped, with any LLM, including self-hosted models.

xygeni.

Ship artifacts you can prove, not just trust

Attest. Store. Verify. One platform.

Request a PoC | Book a demo

- SLSA provenance in a single CLI step
- Interoperable with cosign, witness and Sigstore
- xygeni.io

RECOGNIZED IN 2026

Hot Company in Application Security Posture Management 2026
Global InfoSec Awards · Cyber Defense Magazine

Some capabilities are in active development and may not be generally available yet; features and timelines are subject to change.

© 2026 Xygeni Security SL.
All rights reserved.