

What a source provenance record can tell you

A practical September introduction to CodeSeal, signed records, and the questions to ask before trusting a verification badge.



By Edwin Rodriguez

Published Oct 14, 2026 · 4 min read

September 2026 retrospective · Story context 2026-09-30



A source archive can move through a surprising number of hands. Someone exports it, someone uploads it, someone renames the folder, and someone else asks whether it matches the version reviewed last week.

A provenance record helps make that conversation more concrete. [CodeSeal](#)'s September work centered on registered source history, fingerprints, and signed evidence. Start with two practical questions: which source does this record describe, and who signed it?

That question is easier to investigate than a broad label such as “verified.” It tells you what to compare and where the remaining uncertainty lives.

| Start with the thing being measured

A fingerprint depends on its input. Hashing a ZIP file includes archive-level details such as file ordering or metadata. Hashing selected source files follows a different rule. Normalizing text before hashing introduces another deliberate choice.

Write the rule down. If two people use different input selection or normalization, different fingerprints can be expected even when the projects look similar in an editor. Conversely, a record that excludes some files says nothing about those excluded files.

For a tiny exercise, create a text file and compute a SHA-256 digest. Change one character and compute it again:

```
from hashlib import sha256
from pathlib import Path

payload = Path("example.txt").read_bytes()
print(sha256(payload).hexdigest())
```

This demonstrates a byte-level fingerprint. A real source registry needs a defined selection process and enough metadata to reproduce the intended comparison.

| Read the signature as a statement

A digital signature lets a verifier check a message using the corresponding public key. The message still needs a clear meaning. It might say that a particular digest was registered at a particular time under a particular identity.

Ask which key is trusted, how the verifier obtained it, and whether the signed bytes match the displayed record. The browser’s cryptographic verification API can answer the signature operation’s mathematical question. Your application still needs to decide which signer it trusts and why.

A valid signature doesn’t establish code quality, a complete security review, or permission to reuse somebody else’s work. Those questions need their own evidence. Keeping the claim precise makes the signed record more useful.

Build a release handoff readers can inspect

For a small application, assemble a release note with the source revision, artifact digest, build method, and the checks performed. Keep the downloadable artifact associated with the same revision the note describes.

Then try a few deliberate changes. Modify a file after creating the record. Substitute a different artifact. Remove a file from the selected input. Check whether the verifier reports the mismatch clearly and whether a reviewer can understand what changed.

These are good negative cases because they resemble ordinary release mistakes. A package can be rebuilt from a newer checkout while the release note still points at yesterday's revision. A checksum copied by hand can belong to the previous download. The tooling should help catch those mismatches before a reader has to.

Keep sensitive source boundaries visible

Some teams need public proof without public source contents. That requires careful separation between a durable evidence record and the underlying material used to create it.

CodeSeal's September work included handling for marked private source sections. A public article can discuss that boundary without exposing any private material. For your own workflow, document what is excluded and test that exclusion with synthetic fixtures. An exclusion feature should never become a vague promise that every possible secret will be detected.

A useful review question

When you see a verification badge, ask for the underlying statement. You want to know the input, the signer, the time, the comparison rule, and the limits.

Use a stable release label so the download, signature, and release note are easy to match. Keep the record beside the release it describes. When somebody asks which version they received, you can compare the artifact, check the signature, and point to the right history. That is a much calmer conversation than trying to remember which ZIP somebody renamed last week.

Sources & further reading

Checked 2026-10-14 · Story context 2026-09-30

1. [CodeSeal](#) (2026-10-14)
2. [MDN cryptographic signature verification](#) (2026-10-14)

[Read the current article online →](#)

Support the workshop

Explore software and learning offers at [StackSocial](#). Affiliate link: ProgreTech may earn a commission from purchases. Check current price, license and renewal terms before buying.

Find our projects at [progretech.com](#).

ProgreTech LLC · Build / Learn / Improve / Together

This file contains its text, illustrations and fonts. No account, scripts or network connection required to read. External reference links need internet access.