PUBLIC DEMONSTRATION // CP-DEMO-SC-01

Artifact integrity and provenance verification

Synthetic security review of a fictional CI/CD verifier

SUBJECT CP-LAB ARTIFACT VERIFIER / FICTIONAL

AUTHOR CYBERPOLAK SECURITY RESEARCH

DATE 18 OCTOBER 2026

SYNTHETIC EXAMPLE - NOT A CLIENT ASSESSMENT

No real repository, product, customer, or production system was tested.

REPORT CONTROL // PUBLIC DEMONSTRATION

Scope, assumptions and method

IMPORTANT: every project name, code path, input, observation and finding in this publication is synthetic. CyberPolak did not inspect a real repository or system for this report.

Report ID	CP-DEMO-SC-01	**Version**	1.0 / English edition
Subject	CP-Lab Artifact Verifier (fictional)	**Prepared by**	CyberPolak Security Research
Date	18 October 2026	**Engagement**	Synthetic demonstration; no client
Target	Constructed CI/CD verifier model	**Evidence**	Invented fixtures; no real capture
Review scope	Signature, subject digest, trust metadata, diagnostics	**Excluded**	Real repos, production, external services

Purpose and boundary

This document demonstrates how a CyberPolak security review can connect a trust decision to its preconditions, impact, remediation and a closure test. The fictional CP-Lab Artifact Verifier represents a small Rust-based CI/CD component that consumes a build artifact and a signed provenance statement.

The sample does not state that the fictional implementation exists. It contains no real customer data, source code, credentials, CVE identifiers or verified findings. The displayed priorities are qualitative labels for the imaginary scenario, not CVSS scores.

Review model

A real review would begin with written authorization and a fixed commit or build. The assessor would trace the verifier's trust decisions, define safe fixtures, compare accepted and rejected cases, preserve minimal evidence, and report uncertainty. Findings would be tied to reproducible behavior and retested after changes.

Technical reference

The terminology in this fictional example uses the SLSA Provenance v1.2 model, including an attestation subject and artifact digest. Referencing a specification does not imply that the imagined tool or this report meets a SLSA level.

```
Reference: https://slsa.dev/spec/v1.2/provenance
```

EXECUTIVE SUMMARY // SYNTHETIC RESULTS

Decision summary

The fictional review model highlights two release-integrity decisions, one trust-metadata lifecycle concern, and one diagnostic-data concern. The counts below exist only to illustrate a report summary; CyberPolak did not find these issues in a real product.

Fictional findings	4 total	Illustrative priority mix	2 high / 1 medium / 1 low
Critical	0	CVE assignments	None; synthetic case
Real systems tested	0	Retest performed	No

Illustrative finding register

ID / PRIORITY	SYNTHETIC SCENARIO	EXAMPLE RESPONSE
CP-DEMO-SC-01 HIGH	Artifact digest not bound to attestation subject	Recompute digest and require exact subject match before release
CP-DEMO-SC-02 HIGH	Required proof unavailable but CI continues	Use explicit verified / rejected / indeterminate states
CP-DEMO-SC-03 MEDIUM	Cached trust metadata freshness not enforced	Validate expiry, version progression and anti-rollback state
CP-DEMO-SC-04 LOW	Diagnostic output may retain a token-bearing URL	Keep secrets out of URLs and redact diagnostic output

How to read the priorities

The labels represent a fictional order of work only. In a real engagement, severity is based on evidence, attacker preconditions, affected assets, business impact and existing controls. An unverified concern is not reported as a confirmed vulnerability.

Decision flow modeled in the sample

```
SOURCE REVISION -> TRUSTED BUILDER -> SIGNED PROVENANCE -> SUBJECT DIGEST MATCH -> EXPLICIT POLICY DECISION
```

Key design question: can the release gate prove that the exact bytes being deployed are the bytes described by an accepted, policy-compliant provenance statement?

DETAILED FINDING // 01 // SYNTHETIC

Artifact digest not bound to the verified attestation subject

QUALITATIVE DEMO PRIORITY: HIGH | NO REAL CODE REVIEW

> SYNTHETIC SCENARIO - This finding is invented for the sample report. It is not attributed to Sigstore, OpenClaw, or any real product.

Scenario

In the fictional flow, a signature and statement structure validate, but the release decision does not compare the digest computed from the downloaded artifact with the digest declared by the statement subject.

Illustrative evidence

A safe demonstration fixture supplies artifact digest A and a signed subject digest B. The imagined verifier returns an accepted result rather than a mismatch. No such source code or execution exists outside this synthetic example.

Potential impact

A release could rely on provenance for a different object from the artifact that is actually promoted. The real impact would depend on how artifacts are fetched, named, cached and deployed.

Recommended remediation

Compute the digest from the exact artifact bytes used by the next stage. Require an algorithm and value match against the verified subject before returning an accepted decision. Keep the verified digest attached to the release record to prevent later substitution.

Closure test

A matching fixture is accepted; mismatched, missing, duplicate or unsupported digest cases are rejected. A regression test covers the artifact bytes at the promotion boundary.

Fictional fixture

```
fixture: statement.subject.digest = sha256:B ; downloaded_artifact.digest = sha256:A ; expected = reject
```

Risk rating is illustrative only. A real assessment would consider the verified implementation, business context, exposure, compensating controls, and authorized scope.

DETAILED FINDING // 02 // SYNTHETIC

Unavailable required proof is treated as a warning

QUALITATIVE DEMO PRIORITY: HIGH | NO REAL CODE REVIEW

> SYNTHETIC SCENARIO - This finding is invented for the sample report. It is not attributed to Sigstore, OpenClaw, or any real product.

Scenario

The imaginary CI policy continues publication when a required transparency proof cannot be fetched or verified, recording a warning but returning the same success state as a verified release.

Illustrative evidence

A controlled synthetic fixture makes the proof source unavailable. The model returns a successful release result with a warning. The report invents this behavior; no live service or repository was queried.

Potential impact

Operators may interpret a warning as a successful verification. A transient outage could therefore weaken the release gate without an explicit policy decision.

Recommended remediation

Represent verification outcomes as distinct states such as verified, rejected and indeterminate. For protected releases, block on indeterminate unless an authorized offline mode validates a complete local proof bundle under an explicit policy.

Closure test

A verified proof is accepted; invalid proof is rejected; unavailable required proof yields indeterminate and does not publish. Offline acceptance requires explicit policy and complete evidence.

Fictional fixture

```
fixture: required_proof = unavailable ; policy = protected_release ; expected = no_publish
```

Risk rating is illustrative only. A real assessment would consider the verified implementation, business context, exposure, compensating controls, and authorized scope.

DETAILED FINDING // 03 // SYNTHETIC

Cached trust metadata lacks freshness and anti-rollback checks

QUALITATIVE DEMO PRIORITY: MEDIUM | NO REAL CODE REVIEW

SYNTHETIC SCENARIO - This finding is invented for the sample report. It is not attributed to Sigstore, OpenClaw, or any real product.

Scenario

The fictional cache accepts signed trust metadata without enforcing expiry or monotonic version progression after a key or trust-root rotation.

Illustrative evidence

A constructed test vector presents an older, still correctly signed metadata version after a newer version has been observed. The imagined verifier accepts the stale version. This is not a claim about any real verifier.

Potential impact

After trust material changes, a consumer could continue using a superseded policy longer than intended. Exploitability depends on the update and cache model.

Recommended remediation

Validate expiration and version progression, retain the minimum accepted version, and define recovery for unavailable updates. Make cache state integrity and rollback behavior explicit.

Closure test

Expired, downgraded and out-of-order metadata are rejected. A key-rotation fixture succeeds only when the new trust state is valid and current.

Fictional fixture

```
fixture: cached.version = 7 ; previously_seen.version = 8 ; expected = reject_rollback
```

Risk rating is illustrative only. A real assessment would consider the verified implementation, business context, exposure, compensating controls, and authorized scope.

DETAILED FINDING // 04 // SYNTHETIC

Diagnostic output can retain a token-bearing registry URL

QUALITATIVE DEMO PRIORITY: LOW | NO REAL CODE REVIEW

> SYNTHETIC SCENARIO - This finding is invented for the sample report. It is not attributed to Sigstore, OpenClaw, or any real product.

Scenario

A fictional debug mode logs a complete registry URL that contains a synthetic authentication token in its query string.

Illustrative evidence

A canary-only test fixture inserts a fake token marker and captures standard diagnostic output. The sample assumes the marker remains in the log; there are no real credentials or logs.

Potential impact

If a similar pattern existed in an actual pipeline, logs could expand the audience and retention of a secret. The actual exposure would depend on log access and redaction.

Recommended remediation

Send credentials through a protected header or secret channel rather than a URL. Redact query values and authorization fields in logs, and document log retention and access.

Closure test

A canary token never appears in normal or debug output, including error paths and nested exception messages.

Fictional fixture

```
fixture: registry_url = https://registry.invalid/?token=DEMO-CANARY ; expected_log = token-redacted
```

Risk rating is illustrative only. A real assessment would consider the verified implementation, business context, exposure, compensating controls, and authorized scope.

REMEDIATION AND CLOSURE // DEMONSTRATION

Example remediation roadmap

The table translates each imaginary finding into an owner, a change and evidence that could close it. Real owners and dates must be agreed with the system owner; none are assigned in this demonstration.

ID / PRIORITY	SYNTHETIC SCENARIO	EXAMPLE RESPONSE
CP-DEMO-SC-01 HIGH	Artifact digest not bound to attestation subject	Recompute digest and require exact subject match before release
CP-DEMO-SC-02 HIGH	Required proof unavailable but CI continues	Use explicit verified / rejected / indeterminate states
CP-DEMO-SC-03 MEDIUM	Cached trust metadata freshness not enforced	Validate expiry, version progression and anti-rollback state
CP-DEMO-SC-04 LOW	Diagnostic output may retain a token-bearing URL	Keep secrets out of URLs and redact diagnostic output

Retest protocol

- Record the patched commit, artifact digest and exact policy configuration under review.
- Repeat only the approved fixtures associated with each finding, including negative cases.
- Check for regressions at the build-to-release boundary and retain only sanitized evidence.
- Update each finding as closed, partially addressed, open or not retested; state the retest date and limits.

No closure claims in this sample

No remediation or retest has occurred. The sample intentionally does not say that any issue was fixed, that risk was reduced, or that a product is secure.

About this publication

Prepared by CyberPolak Security Research as an original demonstration of report structure and defensive recommendations. This document is independent and unaffiliated with Sigstore, OpenAI, Trail of Bits, or the maintainers of any real project. Public reports were used only as references for information architecture; their text, findings, code, graphics and identities were not reproduced.

This sample is not a penetration test, audit opinion, certification, legal assessment, or guarantee that a real system is free of vulnerabilities.

`Technical reference: SLSA Provenance v1.2 - https://slsa.dev/spec/v1.2/provenance`