
INDEPENDENT STANDARD SECURITY ASSESSMENT

Peach Browser Independent Standard Security Audit

Report version: 1.0

Assessment status: Final

Classification: Confidential - security-sensitive engineering material

Product baseline: Peach Browser 5.0.27

Koh baseline: bc6649de8443 (dev)

Assessment window: 2026-08-30T12:30Z through 2026-08-30T12:50Z

Assessor: OpenAI Codex, single-agent independent review

Campaign brief: PEACH_BROWSER_INDEPENDENT_AUDIT_CAMPAIGN.pdf, dated 2026-08-29

Executive summary

This standard-depth, whole-repository assessment found no demonstrated path by which an ordinary malicious website, unauthenticated relay peer, or untrusted import could recover the Peach vault root key or bulk plaintext vault data from the assessed local build. One new **Low-severity** release-assurance finding was validated: both Firefox release entry points create a human-readable source archive for Mozilla Add-ons review, but neither submits that archive to `web-ext sign`; the archive is then deleted locally. This reopens the operational objective of the earlier PB-26-004 remediation even though the archive itself now contains the Rust Noise source.

The assessed code contains several strong controls: an exhaustive runtime-message context policy; real-Chromium rejection of content-script bulk-vault access; AES-256-GCM vault protection with bounded parsers; deterministic, byte-identical Noise WASM provenance; small-order X25519 rejection in the JavaScript and Rust paths; signed device membership and terminal revocation handling; bounded locked-sync queues; extension CSP; closed Shadow DOM content UI; and plain-text/sanitized note handling.

The release posture is **conditionally acceptable for local engineering use, but PB-26-018 should be corrected before the next Firefox submission**. This report does not certify a store-distributed binary: no signed CRX/XPI or AMO submission record was supplied. It also does not resolve the owner-accepted plaintext persistence of the long-term sync private key in `chrome.storage.local`.

Severity	Open	Validated
Critical	0	0
High	0	0
Medium	0	0
Low	1	1
Informational / hygiene	4	4

Scope, baseline, and independence

The assessment covered the first-party browser-extension source, Rust Noise core, manifests, build/release automation, tests, documentation, committed generated cryptographic artifacts, and locally rebuilt Chrome and Firefox outputs. The file-level ledger contains 510 in-scope files. Security-critical paths were manually traced; the rest of the repository received automated structural review plus targeted manual inspection. Local caches and third-party installed dependencies were inventoried but not line-reviewed.

Excluded from direct verification were Mozilla/Chrome store infrastructure, production relay and licensing services, mobile-client repositories, browser vendor internals, signed store artifacts, hardware-backed cross-device scenarios, and third-party package source. These exclusions prevent a supply-chain or cross-platform certification.

The work was performed by one assessor, as requested. No separate technical or editorial reviewer participated. Existing audit reports and Koh history were used only to identify regression targets; conclusions were re-established from current source, builds, tests, and tool output.

Environment

Component	Observed
Host	macOS 27.0 (26A5421a), arm64
Node	22.23.1
npm	10.9.8
Bun	1.3.14
Rust / Cargo	1.96.1 / 1.96.1
Playwright	1.59.1
Chromium	150.0.7871.46
Firefox	154.0.1

`package.json` declares Node `>=26.0.0`; the available host used Node 22.23.1. CI passed on that host, but the declared Node 26 environment was not reproduced.

Architecture and security model

Peach is a Manifest V3 password-manager extension with four principal execution zones:

1. Extension pages provide the trusted user interface.
2. A background service worker owns vault keys, cryptography, sync authority, and privileged handlers.
3. Content scripts run in an isolated world on all HTTP(S) pages and request narrowly shaped fill data.
4. Page-main-world WebAuthn relays bridge browser APIs to isolated extension code.

The local vault is encrypted under a random root key. Password/PIN/recovery wrappers protect that root key; vault payloads use authenticated encryption. Sync encrypts entry envelopes from the root key and authenticates devices with deterministic signing and X25519 identities. Noise KK is used for known peers and NK for first pairing. Relay infrastructure is treated as untrusted transport.

Assets

- Vault root key and password/PIN/recovery wrappers.
- Login secrets, TOTP seeds, passkey private material, notes, cards, and history.
- Device signing and Noise private keys.
- Pairing consent, signed membership manifests, revocations, and tombstones.
- License state and release/signing credentials.
- Build provenance for generated WASM and store artifacts.

Principal trust boundaries

- Web page to isolated content script.
- Content script to background runtime messaging.
- Extension page to background runtime messaging.
- Main-world WebAuthn relay to extension relay.
- Locked persistent storage to in-memory unlocked key state.
- Browser to untrusted relay and remote peer.
- Source tree to generated build output and store submission.

Security invariants evaluated

- Websites cannot directly read extension storage or invoke bulk/privileged vault operations.
- A fill response is limited to an origin-authorized entry and strips passkey/TOTP seed/history material.
- Vault plaintext and root-key material are not persisted outside their designed wrappers/session escrow.
- Untrusted imports, sync frames, and relay data are bounded before expensive allocation or parsing.
- Noise sessions authenticate expected devices and reject invalid/small-order keys.
- Revoked devices cannot regain authority by changing an identifier.
- Store artifacts remain traceable to reviewable source, including generated cryptographic code.

Methodology and work-package results

The campaign's WP-00 through WP-10 areas were covered as follows.

WP	Area	Result
WP-00	Baseline and security model	Baseline fixed at Koh <code>bc6649de8443</code> , version 5.0.27; architecture, assets, actors, boundaries, and invariants documented.
WP-01	Extension boundaries and messaging	Exhaustive policy and handlers traced; focused policy/adversarial tests passed; real Chromium rejected content-script storage and bulk-entry access.
WP-02	Vault, keys, lock state, recovery	Encryption, wrapper, PIN, recovery, duress, session escrow, lock, and deletion flows inspected and tested. Accepted session-key and 12-word recovery decisions were not re-raised.
WP-03	Autofill, content UI, passkeys	Origin/frame attribution, visibility/action policy, closed Shadow DOM, WebAuthn relays, RP-ID checks, and secret minimization reviewed.
WP-04	Import/export and parsers	Peach/Codex/CSV/PDF paths, decompression bounds, URL storage allow-list, and PDF sandbox flags reviewed; adversarial parser tests passed.
WP-05	Pairing, Sync, relay	KK/NK flows, SAS consent, signed manifests, revocations, tombstones, queue bounds, and locked spool traced. Long-term locked-sync key persistence remains accepted risk.
WP-06	Cryptography	AES-GCM/KDF usage, RNG, X25519/Ed25519, Noise transcript handling, zero/small-order rejection, and WASM provenance reviewed. Rust tests/provenance passed through CI.
WP-07	Dependencies and third parties	Bun advisory scan and reachability triage completed; Rust dependency tree inventoried. <code>cargo audit</code> was unavailable.
WP-08	History and regression	Koh history and prior PB-26 remediations mapped to current code/tests. PB-26-018 independently demonstrates an incomplete release-path objective.
WP-09	Build and artifact comparison	Chrome/Firefox builds passed; generated manifests compared; Noise WASM hashes matched across source and both builds; no signed store artifact was available.
WP-10	Business controls	Local signed-license handling and privileged message placement reviewed. Remote entitlement service was out of scope.

Static techniques

- File-level repository inventory and classification.
- Manual source-to-sink tracing across runtime messaging, DOM sinks, storage, import/export, sync, pairing, cryptography, and release automation.
- Manifest and built-manifest permission/CSP comparison.
- Searches for dynamic code execution, remote URLs, source maps, logs, suppressions, and secret-bearing shapes.
- Dependency inventory and live advisory scan on 2026-08-30.
- Koh history review without using Git.

Controlled dynamic techniques

- Full standard CI: lockfile freshness, TypeScript, Biome, 3,693 Vitest tests, Noise WASM provenance, Chrome build, Firefox build.
- Focused security/adversarial run: 6 files, 1,451 tests passed.
- Chromium extension run: 26 of 28 tests passed. The two failures were harness defects described below.
- Firefox sender-context runner: bounded environment failure because geckodriver could not reach Marionette on this host.
- SHA-256 comparison of committed and built Noise WASM: all
`14f4ce7f6ad9d6de92ca226f40614a14b045447e3012f85571065e6e32c7fa37`.

Detailed finding

PB-26-018 - Firefox release source archive is created but never submitted

Severity: Low

Confidence: High

CWE: CWE-1104 / release-process assurance weakness (best fit)

Affected: `scripts/release.ts:138-183`, `scripts/release-firefox.ts:53-108`

Security property: Generated/minified Firefox artifacts must be accompanied by reviewable source sufficient to rebuild cryptographic code.

Description

Both Firefox release paths construct `peach-source-${version}.zip` and correctly include `crypto-core`, manifests, application source, build scripts, and lockfiles. Both then invoke `npx web-ext sign` with `--source-dir`, `--artifacts-dir`, and `--channel=listed`, but omit the supported `--upload-source-code` option. Immediately after the signing command, the combined release path calls `rmSync` on the archive and the Firefox-only path moves it to Trash.

The installed `web-ext 10.2.0` help explicitly defines `--upload-source-code` as the path to human-readable source for submissions whose packaged code has been processed. No other submission mechanism references `sourceZip`. Therefore the automated workflow creates the evidence that Mozilla reviewers need but does not attach it to the AMO submission.

Preconditions and exploitability

This is not a direct website-to-vault exploit. It matters when a Firefox release contains transpiled/minified JavaScript or generated WASM requiring source review. The release operator must use one of the two documented scripts. The omission reduces independent review and reproducibility assurance and can cause review failure, delay, or acceptance without the intended source attachment depending on AMO behavior.

Impact

- Mozilla reviewers do not receive the archive through the automated submission path.
- The exact source archive is discarded, weakening artifact-to-source traceability.
- PB-26-004's inclusion of `crypto-core` does not achieve its operational objective.

Evidence

- PB-EV-018: `scripts/release.ts` builds `sourceZip` at lines 138-165, signs without an upload option at 167-181, then deletes it at line 183.
- PB-EV-019: `scripts/release-firefox.ts` repeats the pattern at lines 53-108.
- PB-EV-020: `npx web-ext sign --help` lists `--upload-source-code`; repository search finds no use of the option.

Recommendation

Pass `--upload-source-code ${sourceZip}` to both signing commands. Retain the exact archive (or at minimum its cryptographic digest and immutable build provenance) next to the signed artifact until review completes. Add a dry-run/unit assertion that both release entry points construct a command containing the source upload and that the archive includes `crypto-core/Cargo.toml`, `Cargo.lock`, `src/lib.rs`, and the Noise build script.

Validation procedure

1. Stub the signing subprocess and execute each release entry point in dry-run mode.
2. Assert exactly one `--upload-source-code` argument resolves to the created archive.
3. List the archive and confirm the Rust source/build inputs are present and caches/generated outputs are excluded.
4. Confirm the AMO submission record shows a source-code upload for the version.
5. Verify the retained archive hash is recorded with the signed XPI.

Attack-chain analysis

No complete unauthenticated attack chain to bulk plaintext vault disclosure was validated. The most important attempted chains were:

- **Malicious page -> content script -> privileged runtime handler:** blocked by sender classification for bulk/destructive operations and by browser-attested origin checks plus secret-minimized responses for fill operations. Real Chromium confirmed `storage.local` and `GET_ENTRIES` rejection from a content script.
- **Malicious iframe -> top-frame authority confusion -> cross-origin credential:** mitigated by preferring `sender.url` (frame URL) over `sender.tab.url`, extracting registrable domains, and requiring explicit user summon for off-domain fill.
- **Relay/peer -> oversized frame/import -> memory or parser exhaustion:** bounded by frame, entry, tombstone, declared decompression, and import limits; focused bomb/adversarial tests passed.
- **Rogue sync peer -> identity substitution:** mitigated by Noise remote-static authentication, signed device manifests, expected-key checks, terminal revocations, and small-order key rejection.
- **Build compromise -> opaque generated cryptography:** local CI rebuilt the Noise core byte-identically, but PB-26-018 leaves the external Firefox review handoff incomplete.

Verified controls

The following controls were directly supported by code inspection plus tests or build evidence:

- Runtime message types are exhaustively classified and unknown/background-internal types fail closed.
- Bulk vault, recovery, destructive, pairing-authority, and device operations are extension-page-only.
- Fill responses omit passkey material, raw TOTP seeds, and password history; origin mismatch returns a non-oracular not-found response.
- Storage access is restricted to trusted extension contexts where the browser supports `setAccessLevel`.
- Vault/import payloads use authenticated encryption and bounded decoding/decompression.
- Note paste is plain text and note HTML conversion strips attributes and non-allowlisted elements.
- Content UI uses closed Shadow DOM and the extension has no `externally_connectable` surface.
- Noise core provenance is enforced in CI and the committed/built WASM hashes match.
- JavaScript and Rust Noise paths reject known small-order X25519 public keys.
- Sync manifests are signed; long-term key revocation is terminal; tombstones survive purge in the encrypted ledger.

These statements apply only to the inspected baseline and are not a general guarantee of security.

Observations, hygiene, and unresolved risks

OBS-01 - Dependency advisory debt

`bun audit` reported 38 advisories (2 critical, 24 high, 9 moderate, 3 low) across 13 package names. Most are in developer/release tooling (`web-ext`, Vite, PostCSS, Babel, jsdom) or APIs not found in first-party runtime calls; no reachable packaged-extension exploit was demonstrated. `solid-js` brings the critical `seroval` advisory, but repository/bundle searches found no first-party `seroval.fromJSON()` use. Treat the counts as maintenance debt, not product severity. Upgrade with regression testing and rerun the scan. The advisory snapshot is time-sensitive as of 2026-08-30.

OBS-02 - Browser E2E harness defects

The installed-Chromium run passed 26/28 tests. `SECS02` performs a nested Playwright launch that does not honor `PLAYWRIGHT_CHROMIUM_EXECUTABLE_PATH`, and `SECE02` searches for an obsolete `aes`-named bundle rather than proving WebCrypto behavior. These failures do not negate the separately inspected/session-tested controls, but they reduce independent browser evidence. Repair the harness and keep environment failures distinct from assertions.

OBS-03 - Declared toolchain mismatch

The project requires Node ≥ 26 , while this assessment host provided Node 22.23.1. Standard CI nevertheless passed. Re-run release qualification under the declared Node 26 toolchain and pin it in CI to avoid silent compatibility drift.

OBS-04 - Cleanup policy inconsistency

`scripts/release.ts` uses `rmSync` for generated archives while workspace policy requires recoverable Trash operations. This is not a security vulnerability by itself, but it compounds PB-26-018 by destroying the only local copy of the source-review archive.

Accepted residual risk - locked sync private key

`src/background/locked-sync-spool.ts` persists `longTermPrivateKeyHex` in plaintext in `chrome.storage.local` so locked sync can continue without the root key. This is explicitly accepted and deferred in repository policy. Compromise of extension-context storage could enable persistent mesh impersonation and traffic observation, but not vault plaintext decryption by itself. Revisit before wider release.

Sender-metadata limitation

Extension-page-only policy intentionally accepts URL-less same-extension senders, and full-entry popup flows skip domain checks when no tab/URL exists. Browser APIs supply sender metadata for normal content scripts, and the real Chromium boundary test passed. Firefox's equivalent sender-context check could not run because the local Firefox process never exposed Marionette. No exploit was established; retain a real-Firefox CI sentinel.

Test limitations and contradictory evidence

- No signed CRX/XPI, store metadata, or production submission record was available. Only local build outputs were compared.
- Firefox 154.0.1 launched outside WebDriver, but its plugin-container/Marionette path did not become ready under geckodriver 0.37.1 within 30 seconds.
- The Playwright bundle for version 1.59.1 was not installed. Most tests were rerun with installed Chromium 150; one nested launch ignored that override.
- `cargo audit` was unavailable and was not installed; Rust dependencies were inventoried and the Rust tests/provenance build ran through CI.
- `npm audit` is inapplicable because this Bun-lock project has no `package-lock.json`; `bun audit` was used.

- Production relay, license, and browser-store services were not exercised.
- Cross-platform protocol behavior was inferred from Browser code and frozen vectors only; mobile clients were outside scope.
- Single-agent execution means no independent second review was performed.

Conclusion

At standard depth, Peach Browser 5.0.27 presents a mature defensive design with strong local vault, message-boundary, parser, and sync controls. The assessment found no Critical, High, or Medium issue and one High-confidence Low issue in the Firefox release handoff. Fixing PB-26-018 before the next AMO submission, repairing the two E2E harness defects, and qualifying under Node 26 would materially improve release assurance. The accepted locked-sync private-key persistence remains the largest documented residual architectural risk.

Appendices

A. Evidence and support package

- `peach-browser-standard-audit-2026-08-30/coverage-ledger.csv`
- `peach-browser-standard-audit-2026-08-30/evidence-index.md`
- `peach-browser-standard-audit-2026-08-30/finding-register.json`
- `peach-browser-standard-audit-2026-08-30/test-record.md`
- `peach-browser-standard-audit-2026-08-30/architecture.mmd`

B. Exact command summary

Commands, results, limitations, and environment details are recorded in `test-record.md`. All repository history and save operations used Koh; Git was not used.

C. Severity and confidence

Severity reflects demonstrated impact and realistic prerequisites: Critical (systemic secret compromise), High (major secret/authority compromise), Medium (material but constrained compromise), Low (limited security or assurance impact), Informational (hardening/maintenance). Confidence is High when source plus executable/tool evidence establishes the behavior, Medium when evidence is strong but environment-dependent, and Low when a plausible concern lacks a demonstrated path.