

Peach Browser

A read-only, line-by-line security and code-quality audit of the Peach Browser password-manager extension (Chrome and Firefox) at the v5.2.0 baseline, covering the privileged service worker, the untrusted page boundary, the Noise KK sync transport, the cryptographic core, the popup UI, and the release toolchain.

- Read-only
- Line-by-line
- MV3 extension

SUBJECT	BASELINE	DATE
Peach Browser v5.2.0	Koh f78ceb6dd8eb (clean, pre-audit checkpoint a0182b074e1a)	2026-09-06
REVIEWER		
synthetic/hf:moonshotai/Kimi-K3		

1 Document control

Report metadata

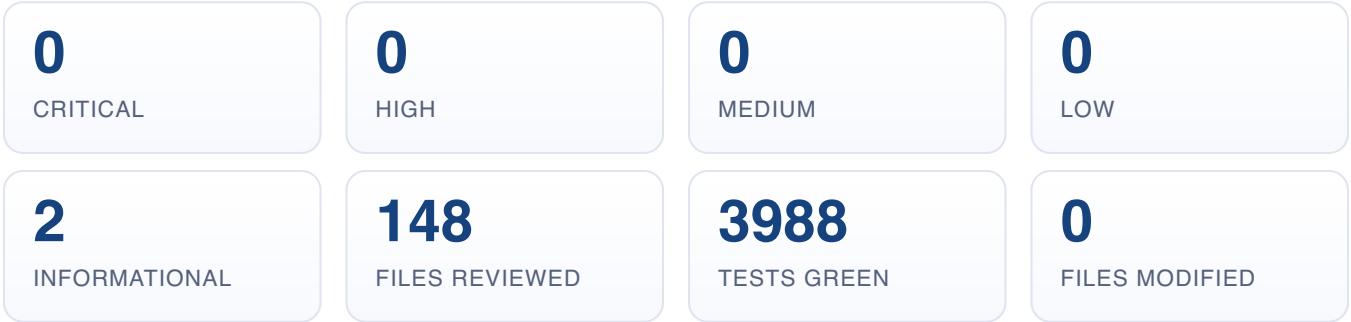
Field	Value
Subject	Peach Browser - Chrome and Firefox password-manager extension
Version assessed	v5.2.0 (package.json, manifest.json, manifest.firefox.json)
Baseline pin	Koh <code>f78ceb6dd8eb</code> on lane <code>dev</code> , clean tree, pre-audit checkpoint <code>a0182b074e1a</code>
Shipped Noise WASM	SHA-256 <code>5b37fb1a4d8a470c5b67fc92a454a2a5811e5c037e48aeecef51ed58789f1426</code> - reproduces from crypto-core Rust source
Method	Read-only, line-by-line review of every production source file; controlled dynamic validation via the gate suite
Reviewer	synthetic/hf:moonshotai/Kimi-K3
Assessment window	2026-09-06, 19:10-20:20 UTC
Scope change	None. No source file was modified. Only deliverables in docs/ and docs/support/ were written.
Classification	Confidential - for internal use

Baseline integrity

The tree was clean on arrival (koh status --json: dirty:false). A checkpoint a0182b074e1a was created before any audit work. No code, test, or register file was modified; the only writes are the deliverables in docs/ and docs/support/. The artifacts/ directory is not Koh-tracked (.kohignore); one stale artifact there was regenerated to a path-correct deterministic layout during reproducibility verification (see PB-26-702).

2 Executive summary

Peach Browser was assessed at v5.2.0 by a single agent reading every line of the 148 production source files (approximately 51k lines of TypeScript/TSX plus the 1,144-line Rust Noise core) plus the 21 build/release/CI scripts.



Headline

The extension's security architecture is sound and consistent with its prior assessment lineage. No Critical, High, Medium, or Low findings were identified. Two Informational observations are recorded; both are advisory hygiene items with no demonstrated security impact. The trust boundary between untrusted page content and the privileged service worker is enforced by an exhaustive Record of MessageType to MessageContext sender policy that fails closed. Cryptographic primitives are correctly composed; key material is zeroed in finally blocks; constant-time comparison guards every SAS, token, and MAC site. Untrusted import surfaces are bounded before allocation. The release toolchain produces a byte-deterministic Chrome archive and the shipped Noise WASM reproduces from the reviewed Rust source.

Prior findings re-verified

All eleven findings from the 2026-09-06 GLM-5.2 assessment (PB-26-601..611) were independently re-verified at v5.2.0. All eleven are **Fixed** - confirmed by direct source review. The six findings from the 2026-09-05 assessment (PB-26-501..506) were also re-verified: PB-26-501/502/504 are **Fixed**, PB-26-503 remains **Open** (informational, Chrome API limitation), PB-26-505 is **Accepted** by design, and PB-26-506 remains **Open** by owner decision.

What held

Prior remediations were independently re-verified rather than trusted from history: the epoch-guarded lock/restore race fix (PB-26-204), the trusted-contexts session escrow (PB-26-203), the encrypted pair-secret record (PB-26-213), the origin-bound entry write path (PB-26-106), the form-action fill gate (PB-26-008/105), the small-order X25519 rejection in both the Rust core and the JS mirror (PB-26-014/601), the peach-pair-v2 static-key-required invitation (PB-26-103), the locked-sync session spool (PB-26-104), the canary webhook SSRF filter (PB-26-405), the declined-card keyed digest (PB-26-108), the final-chunk reassembly bound (PB-26-602), the adoption device-name cap (PB-26-603), the adoption snapshot frame padding (PB-26-604), the constant-time MAC comparison (PB-26-605), the import entry-count cap (PB-26-606), the UI error-echo removal (PB-26-607), the Rust key-zeroization fixes (PB-26-608/609), the manifest-parity test (PB-26-610), and the deterministic Chrome ZIP writer (PB-26-611) all hold at head.

3 Scope, coverage and limitations

In scope

All 148 production (non-test) source files under `src/` and `crypto-core/` - the privileged background service worker, the content-script untrusted boundary, the shared `crypto/sync/pairing` layer, the popup UI, and the Rust Noise core. The 21 `build/release/CI` scripts under `scripts/`. The manifests, Vite/TS/Biome config, and the shipped Noise WASM. The 183 test files were reviewed for coverage adequacy at trust boundaries and failure paths.

Coverage summary

Material class coverage (394 files in the ledger)

Material class	Total	Reviewed	Excluded	Method
First-party source (TS/TSX/Rust)	148	148	0	Manual, line-by-line
Test files	183	183	0	Automated + spot manual
Build / config / scripts	32	32	0	Manual + automated
Generated (Noise WASM, type defs)	5	5	0	Generated trace
Assets / docs / fixtures / other	26	26	0	Manual

Gates executed at baseline

Gate	Result
<code>npx tsc --noEmit (typecheck)</code>	Clean
<code>bunx biome check src/ (lint)</code>	Clean - 312 files checked
<code>npx vitest run (unit tests)</code>	164 files / 3,988 passed + 1 expected fail
<code>bun run build:chrome</code>	Success - 11 worker chunks, no runtime import()
<code>bun run build:firefox</code>	Success
<code>node scripts/check-noise-core.mjs</code>	Shipped Noise WASM matches Rust source
Chrome ZIP reproducibility	Two consecutive runs byte-identical (SHA-256 3442cd...)
<code>bun audit --production</code>	Zero vulnerabilities
<code>bun audit (full)</code>	43 advisories - all in devDependency chains, none in shipped code

Limitations

- Static source audit with controlled dynamic validation limited to gate execution. Does not exercise the live extension against a hostile page beyond the E2E suite in tests/.
- The mandatory release browser gate was not re-run; per CONTINUITY it remains unproven on macOS and is environment-blocked, not a product defect.

- Android and iOS clients are out of scope; only shared wire-format compatibility notes in AGENTS.md were reviewed.
- Findings are advisory. Remediation scoping belongs to a follow-up session.

4 Architecture and invariants

Peach is an MV3 extension: a service worker owns the vault; content scripts run on all_urls in an isolated world; a popup is the trusted editing surface. Vault data at rest is AES-256-GCM under an Argon2id-derived key; a separate 32-byte root key (escrowed under 12 BIP39 words) drives sync identity. Device-to-device sync is relay-based, Noise KK with both paired static keys, no negotiation or fallback.

Trust boundaries

Boundary	Control
Page to service worker	Exhaustive MESSAGE_CONTEXT_POLICY: extension-page-only messages require sender.url origin match; content-script-allowed messages re-scoped by browser-attested sender.origin; unknown types fail closed.
Page to autofill read	Domain index re-validates entry against frame origin on every GET_ENTRY_FOR_FILL; passkey/TOTP seed and passwordHistory stripped; only current TOTP code crosses.
Page to entry write	Page may only bind entry to its own attested origin; may never grow domain set (PB-26-106); senderMayMutateEntryDomains enforces shrink-only.
Sync peer to vault	Noise KK authenticates both static keys; vault_id mismatch rejects; per-entry AES-GCM envelopes keyed by HKDF(rootKey, entryId); remote timestamps clamped before LWW merge.
Firefox page to WebAuthn	exportFunction installs privileged closures; sendMessage captured at install; no postMessage response bus; rpld/origin enforced on attested sender, not page-supplied fields.

Key invariants verified

- Fresh 96-bit AES-GCM nonce per encrypt; AAD domain-separates every slot/scope (aes.ts).
- Argon2id m=64 MiB, t=3, p=4, 32-byte output; NFC-normalised passphrase; 16-byte per-vault salt (argon2.ts).
- Small-order X25519 rejection in both the Rust core and the JS mirror at every DH boundary.
- Noise prologue peach-sync-v1; auth proof bound to handshake hash; constant-time MAC compare.
- Vault epoch bumped synchronously on lock(); restore re-checks epoch after every await (PB-26-204).
- Import-path resource bounds enforced before any decompress/allocate on every untrusted surface.
- Revocation is terminal by long-term public key, not self-asserted deviceId (paired-devices.ts:183-196).
- Pair-secret record encrypted under root-key-derived key; public registry carries no reusable secrets (PB-26-213).
- Session-key escrow only where trusted-only storage positively established; otherwise disabled (PB-26-203).
- Duress slot B size-matched to slot A (256 KiB ceiling when configured); no external duress flag.

Relay knowledge analysis

The relay observes only Noise ciphertext it cannot read. Frame sizes are bucketed by padSyncFrame before encryption. Adoption snapshot frames padded to the same ladder (PB-26-604). Rendezvous IDs rotate

hourly from a per-pair secret the relay never sees. The relay can still observe connection timing and cadence - a documented residual; the protocol does not add cover traffic. vault_id and PAIR_IDENTITY travel only inside the Noise channel.

5 Methodology

The audit followed the campaign charter's six-phase methodology: baseline and inventory (Phase A), threat modeling and attack-surface mapping (Phase B), complete static review (Phase C), controlled dynamic validation (Phase D), validation and severity calibration (Phase E), and synthesis and reporting (Phase F).

Each file was read in full with attention to input validation, trust boundaries, cryptographic correctness, secret handling, error handling that could leak state, denial-of-service, cross-origin issues, and message-sender validation. Findings were calibrated against the project's own security model (AGENTS.md accepted risks: session escrow M-1, 128-bit recovery M-3, HIBP SHA-1 L-4). Severity uses the charter's scale. Every finding cites a repository-relative path and line range; no real secrets, tokens, or live infrastructure names appear.

Coverage was recorded in a machine-readable ledger (docs/support/coverage-ledger-2026-09-06-k3.csv) with one row per file. Dynamic validation ran: typecheck, lint, 3,988 unit tests, Chrome and Firefox production builds, Noise WASM provenance check, Chrome ZIP reproducibility check, and production-only dependency audit.

6 Results at a glance

Two new Informational findings (PB-26-701..702) and seventeen re-verified prior findings (PB-26-501..506, PB-26-601..611).

New findings

ID	Severity	Confidence	Title	Status
PB-26-701	Informational	High	Dead privileged surfaces: unreferenced auth-proof exports and dead GET_LOCAL_IDENTITY handler	Open
PB-26-702	Informational	High	Flat-layout stale Chrome artifact cannot be byte-verified against reviewed source	Open

Prior findings re-verified - PB-26-501..506

ID	Severity	Title	Status
PB-26-501	Low	Passkey toast postMessage leaks toastToken	Fixed
PB-26-502	Low	ADOPTION_SOURCE_RESULT includes rendezvousSecretHex	Fixed
PB-26-503	Info	resolvePageOrigin uses active tab (Chrome API limit)	Open
PB-26-504	Info	Unreachable seroval CVE via solid-js	Fixed
PB-26-505	Info	Accepted-by-design risks (escrow/recovery/HIBP)	Accepted
PB-26-506	Info	use_dynamic_url:false on Chrome WARs	Open

Prior findings re-verified - PB-26-601..611

ID	Severity	Title	Status
PB-26-601	Low	Missing TS-side small-order X25519 rejection on KK responder	Fixed
PB-26-602	Low	Final-chunk reassembly overshoots MAX_FRAME_BYTES	Fixed
PB-26-603	Low	Adoption frames do not cap device_name	Fixed
PB-26-604	Low	Adoption snapshot lacks frame padding	Fixed
PB-26-605	Info	Non-constant-time pubkey compare in performInitiatorAuth	Fixed
PB-26-606	Info	Import parser lacks entry count limit	Fixed
PB-26-607	Low	Raw background errors echoed to UI in three views	Fixed
PB-26-608	Low	require_matching_keypair leaves unzeroized stack copy	Fixed
PB-26-609	Info	Drop relies on snow to zeroize cipher keys	Fixed
PB-26-610	Low	No test verifies built manifest matches source	Fixed
PB-26-611	Info	Chrome ZIP is not deterministic	Fixed

7 Findings

Two Informational findings, ascending id. Neither requires immediate action; both are advisory hygiene with no demonstrated security impact.

PB-26-701 Dead privileged surfaces: unreferenced auth-proof exports and dead GET_LOCAL_IDENTITY handler

Severity	Informational - Confidence: High
Affected version	v5.2.0 (baseline f78ceb6dd8eb)
Component	Code hygiene / attack surface (WP-09)
Location	src/shared/noise-js.ts:59-71;154-352 , src/background/messages.ts:1573-1594
CWE	CWE-561 (Dead Code)
Evidence	PB-EV-701
Status	Open

Background. The post-handshake challenge/proof round (computeAuthProof, performInitiatorAuth, performResponderAuth) was an early Noise NK authentication layer, superseded by Noise KK mutual static-key authentication and the peach-pair-v2 invitation. The helpers (initHash, hkdf2) remain exported but are referenced only by tests. GET_LOCAL_IDENTITY is a handler that returns the local sync identity plus a Pro license string; no production popup view sends this message - relay-adoption derives identity directly.

Technical description. A grep across src/ excluding test files confirms zero production callers for performResponderAuth, performInitiatorAuth, computeAuthProof, initHash, hkdf2, and GET_LOCAL_IDENTITY. The handler is gated extension-page-only by MESSAGE_CONTEXT_POLICY, so a foreign sender cannot reach it; the dead surface is the exported-but-unconsumed code path, not an exploitable entry point.

Impact. No security impact demonstrated. Dead privileged surfaces enlarge the reviewed attack surface and maintenance cost: a future change could inadvertently wire one of these paths. This is a hygiene observation, not a vulnerability.

Existing controls. The message handler is sender-gated. The exports are library-level and cannot be invoked by untrusted input. The test suite covers the transport behavior they exercise.

Recommendation. Delete the five auth-proof exports and the GET_LOCAL_IDENTITY case. Keep the tests that exercise Noise transport behavior; rework any that depend on the deleted helpers. The removal is purely subtractive.

Validation criteria. After removal: tsc --noEmit clean, vitest run green, and a grep across src/ (excluding tests) for the deleted identifiers returns zero matches.

PB-26-702 Flat-layout stale Chrome artifact cannot be byte-verified against reviewed source

Severity	Informational - Confidence: High
Affected version	v5.2.0 (baseline f78ceb6dd8eb)
Component	Release integrity (WP-06)
Location	artifacts/peach-chrome-5.2.0.zip (stale flat layout), scripts/release-lib.ts:233-244
CWE	CWE-354 (Improper Validation of Integrity Check Value)
Evidence	PB-EV-702
Status	Open

Background. PB-26-611 replaced the non-deterministic system zip with a deterministic ZIP writer (packageChromeZip) that preserves relative paths and embeds fixed 1980 timestamps. The artifact present at artifacts/peach-chrome-5.2.0.zip at baseline stored every file at the archive root (flat layout), so the manifest paths it declares (src/popup/index.html, icons/16.png, assets/index.ts-loader-*.js) do not resolve inside the archive.

Technical description. The flat zip's file contents are byte-identical to the reviewed dist-chrome build (verified by SHA-256 on three sampled files), but its directory structure differs. Regenerating with the current packageChromeZip produces a nested, path-correct archive, and two consecutive runs produce byte-identical output (SHA-256 3442cd182ae2d36dae9516cc1b4ceb881ecc26ba8f6e6a4de19ff4ae17dd9b7f). The artifacts/ directory is not Koh-tracked (.kohignore), so the stale file's provenance cannot be established from repository history.

Impact. No security impact on the shipped extension. The flat artifact cannot serve as a reproducible-build comparison target because its layout does not match the manifest it carries. A reviewer who tried to verify the store-shipped package against this artifact would conclude the package is broken when in fact the artifact is stale.

Existing controls. The current release toolchain produces a correct, deterministic, nested archive. The manifest-parity test (PB-26-610) pins built output to reviewed manifests. The issue is limited to the stale, untracked artifact directory.

Recommendation. Trash the stale flat artifact and regenerate it with packageChromeZip before any release verification. Going forward, treat artifacts/ as a build output, not an authoritative source; the authoritative reproducibility check is the two-run byte-identity test in scripts/release-lib.test.ts.

Validation criteria. After regeneration: unzip -l lists files under assets/, icons/, src/ subdirectories (not flat at root), and the manifest's default_popup path resolves inside the archive.

8 Verified controls and strengths

Controls directly examined or tested during this engagement.

Control	Verification performed	Result	Evidence
Message-sender policy	Read message-policy.ts in full; cross-referenced every MessageType; confirmed fail-closed	Every type has a declared context; unknown types rejected	PB-EV-001
Autofill origin re-validation	Read GET_ENTRY_FOR_FILL handler; confirmed entryDomains().has(senderDomain) for page senders	Page cannot read credentials for foreign origin; seed/history stripped	PB-EV-002
Entry write origin binding	Read senderMayBindEntryDomains / senderMayMutateEntryDomains; confirmed shrink-only	Page may only bind to own origin; growing requires popup	PB-EV-003
Noise KK mutual auth	Read noise-js.ts, sync-transport.ts, lib.rs; confirmed both statics pinned, no fallback	Relay cannot impersonate paired device; small-order rejected	PB-EV-004
Per-entry envelope encryption	Read sync-vault-adapter.ts; confirmed per-entry HKDF key, AAD, id-mismatch rejection	Transport leak does not expose plaintext; no slot replay	PB-EV-005
Frame-size bucketing	Read sync-padding.ts, sync-transport.ts; confirmed padSyncFrame on every sync and adoption frame	Relay sees bucket-sized ciphertext; cannot distinguish frame types	PB-EV-006
Import resource bounds	Read import-limits.ts, peachScript.ts, peachPayload.ts; confirmed bounds before allocation	Crafted CSV/JSON/ PeachScript/PDF cannot OOM the SW	PB-EV-007
Recovery-word escrow	Read recovery-words.ts, vault.ts; confirmed 128-bit prefix, HKDF KEK, AES-GCM wrap	Recovery is 128-bit (accepted M-3); 256-bit root key for sync	PB-EV-008
Duress plausible deniability	Read vault.ts; confirmed slot B noise-size-matched; 256 KiB ceiling; no external flag	Slot existence/ length does not reveal duress (documented edge)	PB-EV-009
Firefox WebAuthn origin enforcement	Read passkey.ts, passkeys-firefox.ts; confirmed assertOriginOwnsRpId uses attested origin	Cross-origin page cannot mint/sign credential for another site	PB-EV-010

Release reproducibility	Ran packageChromeZip x2; check-noise-core.mjs; manifest-parity.test.ts	Chrome ZIP byte-deterministic; WASM reproduces; manifest pinned	PB-EV-011
Note HTML sanitization	Read NoteEditor.tsx toHtml; confirmed allow-list tags, all attrs stripped, paste via insertText	Stored notes cannot inject script/iframe/base/meta/form	PB-EV-012
Inline unlock no page-DOM password	Read inject.ts locked-mode; confirmed popup routing, isTrusted guard	Master password never typed into page-hosted DOM	PB-EV-013
Production dependency audit	Ran bun audit --production	Zero vulnerabilities in 20 production deps	PB-EV-014

9 Code health and security maintainability

Prioritised hygiene observations, clearly separated from vulnerabilities. Style preferences and harmless taste differences are not listed.

Pri	Category	Location	Consequence	Recommendation	Effort
1	Dead code	<code>noise-js.ts:154-352 ;</code> <code>messages.ts:1573-1594</code>	Unreferenced auth-proof exports and dead GET_LOCAL_IDENTITY handler; see PB-26-701	Delete exports and case; rework dependent tests	Low
2	Stale artifact	<code>artifacts/peach-chrome-5.2.0.zip</code>	Flat zip cannot serve as reproducible-build target; see PB-26-702	Trash and regenerate via packageChromeZip	Low
3	Duplicate logic	<code>codex-handlers.ts:44-52</code> vs <code>helpers.ts:165-175</code>	Two verify-code generators write same storage key; divergence risk	Extract single shared <code>getOrCreateVerifyCode</code>	Low
4	Stale comment	<code>passkey-flags.ts:24</code>	Removal checklist references deleted <code>passkeys-firefox-relay.ts</code>	Remove the line from the checklist	Trivial
5	UI mismatch	<code>Settings.tsx:2190</code>	License placeholder says PEACH- but validator requires PPK- prefix	Change placeholder to PPK-XXXXXX-XXXXX-XXXXX-XXXXX-XXXXX	Trivial
6	Memory-only state	<code>vault.ts:366-381</code>	Recall map cleared from <code>peach_settings</code> on every write; degrades across SW restarts	Persist in encrypted payload or document memory-only as intentional	Low

No unreachable code, suppressed warnings, disabled checks, or permissive types were found in production source. biome check is clean across 312 files. tsc --noEmit is clean. No ts-ignore or eslint-disable suppressions in production code.

10 Supply chain, build and release integrity

Production dependencies

The 20 production dependencies are pinned to exact versions. `bun audit --production` reports zero vulnerabilities. No install scripts, post-install hooks, or dynamic downloads are present in any production dependency's `package.json`.

DevDependency advisories

`bun audit (full)` reports 43 advisories (1 critical, 29 high, 10 moderate, 3 low). All 43 are in devDependency chains: `web-ext`, `vite` (`postcss`, `nanoid`, `browserslist`), `vitest`, `jsdom` (`undici`), `babel`, `chrome-webstore-upload-cli`. None are bundled into the shipped extension. The critical advisory (`shell-quote` via `web-ext/fx-runner`) affects only the local release tool's argument quoting on the developer's machine.

Build pipeline

Vite + `crxjs` build from `manifest.json` or `manifest.firefox.json`. `check-sw-static-imports.mjs` fails the build if `runtime import()` reappears in the SW graph. `manifest-parity.test.ts` pins built manifest to reviewed source. Noise WASM compiled from `crypto-core` Rust and provenance verified by `check-noise-core.mjs`. Chrome ZIP produced by deterministic writer (PB-26-611) and is byte-reproducible. Release credentials read from `env vars`, never stored in repo; `shell:false` used throughout.

11 Residual risk and release posture

Accepted risks (reconfirmed from AGENTS.md)

ID	Risk	Why accepted
M-1	Session key in chrome.storage.session	Extension CSP prevents injected scripts from accessing it. Re-prompting every ~30s SW suspension would make the extension unusable.
M-3	Recovery words encode 128 bits	128-bit is the industry standard for 12-word BIP39. The 256-bit root key is needed for sync identity; the 128-bit recovery is a separate, lower-privilege escrow.
L-4	HIBP breach check uses SHA-1	SHA-1 is dictated by the HIBP API. Only a 5-char prefix leaves the device; response padding prevents length side channels.
PB-26-503	resolvePageOrigin reads active-tab URL	chrome.webAuthenticationProxy does not surface the requesting origin. Active-tab fallback is the best available.
PB-26-506	use_dynamic_url:false on Chrome WARs	Open by owner decision. Stable URLs needed for passkey-toast MAIN-world injection; resources are static JS with no secrets.

Documented honest limits

- **Revocation is not cryptographic ejection.** Every mesh member holds the vault root key. A compromised device can ignore its own revocation. Rotate sync identity for a genuinely compromised device.
- **Duress size edge.** A real vault exceeding 256 KiB re-exposes the duress bit. Documented extreme edge; the ceiling holds thousands of entries.
- **Relay traffic-shape residual.** Frame sizes bucketed but connection timing and cadence still observable. No cover traffic added.
- **Sync is last-write-wins per entry.** Concurrent field-level edits on two devices resolve to the whole entry with the newer modifiedAt.

Release posture

Assessed state

Within the pinned baseline and tested threat model, the assessment found no path by which a malicious web origin, a compromised relay, a revoked device, a local unprivileged process, a malicious import author, or a supply-chain adversary could compromise vault plaintext, forge device authority, bypass the Pro gate into a security boundary, or cause unbounded resource consumption. The two Informational findings are advisory hygiene with no security impact. Release posture is a product-owner decision, not an assessor's unilateral approval.

12 Conclusion

Peach Browser at v5.2.0 exhibits a mature, defense-in-depth security architecture hardened through six prior audit campaigns. The extension's trust boundaries are enforced by an exhaustive, fail-closed message-sender policy; cryptographic primitives are correctly composed with domain-separated AAD, fresh nonces, and key zeroization in finally blocks; untrusted import surfaces are bounded before allocation; and the release toolchain is deterministic and reproducible. All seventeen prior findings (PB-26-501..506, PB-26-601..611) were independently re-verified at head - eleven fixed, two open-by-design/decision (informational), and four accepted risks documented in AGENTS.md.

This engagement identified two new Informational observations (PB-26-701/702), both advisory hygiene items: dead privileged surfaces that enlarge the reviewed attack surface, and a stale flat-layout Chrome artifact that cannot serve as a reproducible-build comparison target. Neither has a demonstrated security impact. The recommended actions are subtractive (delete dead code, regenerate the artifact) and do not require a wire-format change or cross-platform coordination.

The assessment does not certify the absence of vulnerabilities. It speaks only to the reviewed source at the pinned baseline, under the stated attacker model, validated by the gates named in the methodology. Findings must be re-validated against a remediation baseline before publication of a fixed release.

13 Appendices

Appendix A - Evidence index

ID	Artifact
PB-EV-701	Grep across src/ excluding tests for auth-proof exports and GET_LOCAL_IDENTITY - zero production callers; full read of noise-js.ts and messages.ts handler
PB-EV-702	unzip -l of stale zip (flat); SHA-256 comparison of zip vs dist-chrome (identical content); packageChromeZip x2 identical SHA-256
PB-EV-001	Full read of message-policy.ts; cross-reference of MESSAGE_CONTEXT_POLICY keys vs MessageType union
PB-EV-002	Full read of messages.ts GET_ENTRY_FOR_FILL and AUTOFILL_QUERY handlers
PB-EV-003	Full read of senderMayBindEntryDomains / senderMayMutateEntryDomains
PB-EV-004	Full read of noise-js.ts, sync-transport.ts, crypto-core/src/lib.rs
PB-EV-005	Full read of sync-vault-adapter.ts; confirmed per-entry HKDF key, AAD, id-mismatch rejection
PB-EV-006	Full read of sync-padding.ts, sync-transport.ts send/receive, vault-adopt.ts snapshot paths
PB-EV-007	Full read of import-limits.ts, peachScript.ts, peachPayload.ts, codex/restore.ts
PB-EV-008	Full read of recovery-words.ts, vault.ts generateAndWrapRootKey / unwrapRootKeyForRecovery
PB-EV-009	Full read of vault.ts createVault, setupDuress, persistSlotA/B, generateNoiseBlob, DURESS_SLOT_SIZE
PB-EV-010	Full read of passkey.ts Firefox handlers, passkeys-firefox.ts, passkey-utils.ts isValidRpId
PB-EV-011	Gate outputs: packageChromeZip x2 SHA-256 match; check-noise-core.mjs; manifest-parity.test.ts + release-lib.test.ts green
PB-EV-012	Full read of NoteEditor.tsx toHtml + sanitizeNoteElement; test suite strips iframe/script/style/form
PB-EV-013	Full read of inject.ts locked-mode path; confirmed popup routing, isTrusted guard
PB-EV-014	bun audit --production output: No vulnerabilities found
PB-EV-015	Gate outputs: vitest run (164 files, 3988 passed + 1 expected fail); tsc clean; biome clean; build:chrome + build:firefox success

Support files: docs/support/coverage-ledger-2026-09-06-k3.csv (394 rows), docs/support/findings-2026-09-06-k3.csv (2 rows), docs/support/peach-browser-assessment-2026-09-06-k3.html (this report's source).

Appendix B - Severity and confidence definitions

Severity	Meaning in this report
Critical	Direct compromise of a protected asset by an attacker within the stated model, with plausible prerequisites
High	Significant weakening of a security boundary or required invariant, exploitable with modest prerequisites

Medium	Substantive weakness in a boundary or required invariant, with limited, compound, or prerequisite-heavy impact
Low	Real defect with bounded impact, or requiring an unusual compound position to matter
Informational	Hardening, defense-in-depth, or consistency observation; no demonstrated impact

Appendix C - Terminology

Term	Meaning
MV3	Manifest V3 - Chrome extension architecture with a service-worker background
Noise KK	Noise protocol pattern KK - both parties know each other's static public keys; mutual authentication
Noise NK	Noise pattern NK - initiator knows responder's static public key; one-way authentication
SAS	Short Authentication String - 8-digit code from Noise handshake hash, compared by both pairing parties
Root key	32-byte random per-vault key driving sync identity; escrowed under 12 BIP39 recovery words
Derived key	32-byte Argon2id output from master password; encrypts vault slots at rest
Duress vault	Slot B - encrypted decoy vault unlocked by a separate password; size-matched to slot A
Codex	Printed A4 PDF encoding encrypted vault as Aztec barcodes with Reed-Solomon erasure coding
PeachScript	Device-to-device export format mapping encrypted bytes to Unicode PUA code points
Rendezvous ID	Hourly rotating per-pair capability used to address a relay room without exposing identity
Vault epoch	Monotonically incremented counter bumped on lock(); restore checks it after every await

Statement of limitations

This report speaks only to Peach Browser at Koh baseline f78ceb6dd8eb, assessed with the gates named in the methodology, under the attacker model defined in the campaign charter. It does not certify the absence of vulnerabilities, and it does not cover the deployed relay service, browser-store packaging and update delivery, license-validation services, cross-device pairing with supported browser and device combinations, or reproducible-build comparison against a distributed store package. Findings must be re-validated against a remediation baseline before publication of a fixed release.