

BitTool Signer for Android

Code Security Audit Report

Assessment date: 25 October 2026 (Australia/Sydney)
Assessor: Claude Opus 5.5 (model ID `claude-opus-5-5`), Anthropic — AI-assisted review run in Claude Code
Repository: `github.com/bitbill/bittool-signer-android` (private at the time of review)
Baseline: HEAD `62996b8` (release record only) over code commit `71af693db0c9e728b906c2112e14bab561ed308c`
Version: 1.0.0, version code 1000001; engine `signer.js` SHA-256
`4b3717599e8ee3ed9d2480e640e7203731543cc8ab0f2c3cec3b78e6f400fa19`
Release APK certificate SHA-256: `af41ed101765393ae89038eb4c6fc7fa96d930f6fcd76b881437a39b5a9e4631`
Method: source review of five areas in parallel, lead verification of every reported finding against the code, engine reproductions with synthetic data, dependency and artifact checks. No production code was changed by this assessment.

Disclosure. This report was written by an AI model (Claude Opus 5.5). It is an engineering review of one code snapshot, not an independent third-party audit or certification, and the absence of a finding is not proof that no defect exists. No device testing, malicious-overlay exercise, side-channel analysis or live-chain execution was performed. The same model family also wrote much of the code under review; an independent human audit is recommended before the app protects significant funds.

1. Executive assessment

The wallet's core protections hold up under review. No Critical or High severity issue was found: no password bypass, no way to sign anything other than the reviewed and matched request, no path that extracts the recovery phrase or private keys, no network capability, and no debug code in the release build. The previous findings BTS-01 to BTS-06 are closed in their stated scope (BTS-02 only partially — see BTS-07).

Four Medium findings remain. Two are in the shared signing engine and affect what the user is shown: Solana authority changes can still hide behind a dust transfer (BTS-07), and Bitcoin "change" is accepted at any derivation index, so funds can be parked where watch-only wallets will not look (BTS-08). One concerns the release process: the release signing key and its passwords sit in plaintext in the working tree (BTS-10). The fourth is a mismatch between the stored summary and the review card for native value sent with EVM/TRON contract calls (BTS-09, rated Low-Medium). Nine Low findings and a set of informational notes follow.

Release recommendation: fix BTS-07, BTS-08 and BTS-10 before a general release that protects valuable wallets; the engine fixes belong in the iOS repository (the engine is shared byte-for-byte) and must be re-synced. The Low findings can follow in a point release.

ID	Severity	Area	Finding
BTS-07	Medium	Engine (Solana)	Authority changes, delegations and opaque instructions still hide behind a dust SOL transfer in the headline and stored summary (BTS-02 partly fixed)
BTS-08	Medium	Engine (Bitcoin)	"Verified change" accepted on any profile, branch and index up to $2^{31}-1$
BTS-09	Low-Medium	Engine (EVM/TRON)	Stored summary omits native value sent with an unknown contract call

BTS-10	Medium	Release process	Release signing key and passwords in plaintext, world-readable, in the working tree
BTS-11	Low	Platform	QR camera keeps scanning and importing requests behind the App Lock screen
BTS-12	Low	Store	High-risk acknowledgement enforced only in the UI for fully decoded requests
BTS-13	Low	Crypto	Public-data HMAC key can be regenerated over a live key on first use in a process
BTS-14	Low	Crypto / docs	After dropping <code>setUnlockedDeviceRequired</code> , journal and public-data keys are not gated by the lock-screen check and docs overstate protection
BTS-15	Low	Platform	Default task affinity allows task hijacking on older Android
BTS-16	Low	Setup	Backup verification can be passed by trial and error
BTS-17	Low	Setup	One Recovery Card of a 12-word phrase leaves about 40 bits of entropy
BTS-18	Low	Store	A cancelled match in the keys-locked branch leaves the review on "Finding the matching wallet..."
BTS-19	Low	Platform	Android 8.0–9 cannot detect partial overlays on consent controls

2. Scope and method

The first-party code comprises the Kotlin modules (`app`, `core/*`, `feature/*`), the native host (`core/engine/src/main/cpp`, our JNI over vendored QuickJS 2026-06-04, and the native PBKDF2), and the JavaScript engine (`engine/src`, synced from the iOS repository at commit 5f4f439). The review was split into five areas, each traced end to end by a dedicated reviewer: (1) wallet storage and cryptography; (2) the store and signing pipeline; (3) the engine and its native host; (4) Android platform protections, App Lock and untrusted inputs; (5) setup, recovery and wallet UI flows. The lead reviewer then re-read the cited code for every finding before including it.

Supporting checks on the same commit: the release verification script passed 25 of 25 checks (2,075 JVM/Robolectric unit tests, lint with 0 errors, offline check, 16 KB alignment, permission and component allowlists, engine digest equal to the pin and to iOS); the engine suite passed 217 of 217; `signer.js` rebuilt from a clean `npm ci --ignore-scripts` in a throwaway copy is byte-identical to the shipped file and to the copy inside the APK.

3. Wallet security controls verified

- **Secrets at rest.** PBKDF2-HMAC-SHA256 600,000 iterations (native, ARMv8-accelerated with a runtime capability check, all buffers wiped), AES-256-GCM with fresh 12-byte nonces and a new 32-byte salt per re-encryption, then a per-wallet non-exportable Android Keystore AES-GCM layer (StrongBox when present). Both layers authenticate the exact stored header bytes with domain-separated AAD. Strict bounds on file, header, iterations, salt and ciphertext before any KDF; atomic writes with `fsync` and read-back.

- **Password attempts.** Every path that tests a password (unlock, change, delete, App Lock fallback) goes through one counted vault entry point that persists the attempt before the KDF; a damaged counter fails closed; an attempt is withdrawn only when the device layer fails before the password is tested.
- **Signing integrity.** The plan is bound to the canonical request hash, plan hash and signing-bytes digest; the wallet is matched by public keys, never chosen by hand; the engine re-resolves the plan against the decrypted seed, checks each target key, and rejects any change to the unsigned transaction when accumulating Bitcoin signatures. Session epochs drop late results after backgrounding, App Lock or a new request.
- **Secret lifetime.** Unlocked secrets live in holders closed in `finally`; the derived seed and canonical phrase are owned outside dispatcher switches and wiped even on cancellation (BTS-03 fix intact); no secret in saved state, navigation arguments, logs or `toString`; the phrase is typed on an in-app keyboard with no input-method connection.
- **Chain parsing.** Bitcoin: full previous transactions for non-Taproot inputs, SIGHASH_ALL/DEFAULT only, change re-derived from the wallet's own xpub. EVM: supported chain IDs agreeing across transaction, request and transport; canonical calldata; strict EIP-712. Solana: no lookup tables, signer slots bound, existing signatures verified. TRON: signs the `raw_data` digest, rejects unknown protobuf fields, checks JSON equals the signed bytes.
- **Engine host.** Bundle pinned by SHA-256 and length; runtimes without eval or Function; fresh runtime for every secret- or input-bearing operation; JSON passed as bytes; memory and stack limits; operation allowlist.
- **Platform.** No INTERNET or network-state permission and no networking code; only MainActivity exported, no deep links; backups and device transfer excluded; FLAG_SECURE on every window; foreign overlays hidden on API 31+; tapjacking guards on decisive controls (strict on Sign and the risk acknowledgement); App Lock fails closed, survives process death, uses a monotonic clock, and never authorizes a signature; release logs nothing and contains no debug harness.

4. Findings

BTS-07 — Solana authority changes hide behind a dust transfer (Medium)

Where: `engine/src/solana.js` — System Assign (~287), SPL Approve/ApproveChecked (~293–301), SPL SetAuthority (~337), Metaplex Delegate/Lock (~110–144), Bubblegum delegate (~219–228); headline aggregation (~402–433).

Issue: The BTS-02 fix made Metaplex burns emit "destroys" movements, but authority-changing instructions still contribute only fields and warnings. With a 1-lamport transfer alongside, the transfer is the only movement and becomes the card headline and stored summary.

Reproduced: SetAuthority(AccountOwner → attacker), Approve(u64::MAX → attacker), System Assign and Metaplex Delegate, each with a dust transfer, give summary: "0.000000001 SOL", action "Sign Transaction". An opaque Core BurnV1 plus dust gives action "Transfer SOL". The high-risk warnings remain in the details and risk acknowledgement is required, so this is a consent-presentation defect, not silent signing.

Fix (iOS repo, then sync): give authority-changing decodes a typed effect ranked like burns in the headline and summary; allow a single-movement headline only when every non-ComputeBudget/Memo instruction is that movement; never derive the lone action while an opaque instruction exists; one regression test per opcode.

BTS-08 — Bitcoin change accepted at any index (Medium)

Where: `engine/src/resolver.js` `verifiedBitcoinChange` (~123–139) with `public-index.js` (~87–90).

Issue: An output is "verified change" when its script equals the script derived from any registered single-sig profile, on branch 0 or 1, at any index up to $2^{31}-1$. Re-derivation proves the key is the user's, but not that a watch-only wallet will ever see it.

Reproduced: sending all funds to `m/84'/0'/0'/0/2147483000` (or a legacy path at index 1,999,999,999) reviews as "All outputs return to your wallet / Consolidate". Gap-limited wallets will not discover the funds — the known change-path hiding/ransom pattern.

Fix (iOS repo, then sync): treat an output as change only on the spending input's own profile, branch 1 (optionally 0), below a bounded index (for example the gap limit or $\leq 10^5$); otherwise show it as a payment to "your wallet at an unusual path" with a high-risk warning.

BTS-09 — Stored summary omits native value on unknown contract calls (Low–Medium)

Where: `engine/src/evm.js:97, 342, 352`; `engine/src/tron.js:58–62`.

Issue: " (also sends X)" is appended only when a summary already exists. An unknown payable method carrying 50 ETH is stored as "Unknown contract call"; a token transfer carrying ETH is stored as "Transfer tokens: 1 USDT". The review card and the warning do show the value, so the signing decision is informed; the signature result and Activity history understate it (same class as BTS-04).

Fix: whenever value or call_value is positive on a contract call, build the summary from the same rule as the display ("Sends 50 ETH · <action>").

BTS-10 — Release signing key and passwords in the working tree (Medium)

Where: `keystore/release.jks` and `local.properties` (both mode 0644; git-ignored and absent from history — verified).

Issue: Any code running as the developer — Gradle plugins, the engine's npm dependencies, IDE plugins — can read the key and its passwords. A malicious update signed with this key would install over existing copies, inherit the encrypted vaults and Keystore keys, and show the published fingerprint. Likelihood is low; impact is critical.

Fix: keep the release key off the development machine (hardware token, dedicated release machine or account) and inject it only for the release run via the existing `SIGNER_RELEASE_*` variables; meanwhile `chmod 600` both files; set `SIGNER_RELEASE_CERT_SHA256` so a key swap fails verification; configure a separate Play upload key before the first Play upload.

BTS-11 — QR camera runs behind the App Lock screen (Low)

Where: `core/scanner/.../camera/QrCameraPreview.kt:58` (camera gated on RESUMED only);
`app/.../root/SignerRoot.kt:74` (locked content hidden with alpha);
`feature/sign/.../scanner/ScannerViewModel.kt` (no lock check); the setup recovery-QR scanner follows the same pattern.

Issue: If the camera page is open when App Lock engages, the camera resumes under the lock window and still accepts QR codes; a request shown to the camera is decoded and its review published behind the lock. Signing still needs the wallet password.

Fix: gate the camera on `!access.state.showsLock` and not-captured in addition to RESUMED, or close the scanner when the security epoch changes; drop frames while locked.

BTS-12 — High-risk acknowledgement enforced only in the UI (Low)

Where: `DefaultSignerStore.sign` (~850–880); `engine/src/resolver.js:230` (checks the acknowledgement only for partially decoded or opaque requests).

Issue: For a fully decoded but high-risk request (permit, delegatecall, batch), only the review ViewModel blocks signing without the acknowledgement. Today's UI enforces it correctly; a UI regression or another caller would not be caught.

Fix: in `sign()`, refuse when `review.requiresAcknowledgement && !riskAcknowledged`.

BTS-13 — Public-data key can be regenerated over a live key (Low)

Where: `AndroidKeystoreDeviceKeyStore.publicDataKey` (~157–178).

Issue: The loss proof that guards regeneration runs only once the key has been seen in the current process. On API 26–30, where a failed binder call can report every alias missing, a cold start followed by adding a wallet can generate a new HMAC key over the live one; every existing wallet's public file then fails its MAC and needs Repair. No secret or funds are lost. The code path is confirmed; the platform failure is plausible.

Fix: always require the probe-based proof of absence before generating the key when any vault or public file exists.

BTS-14 — Lock-screen gate and documentation after dropping setUnlockedDeviceRequired (Low)

Where: `KeystoreJournalCipher.seal/open` and `KeystorePublicDataKey.mac/verify` (no `isDeviceSecure()` check); `KDoc` in `AndroidKeystoreDeviceKeyStore`, `DeviceKeyStore`, `KeystoreJournalCipher`, `CryptoModule`, `TestKeystore`; `docs/port/vault-crypto.md`.

Issue: The owner decision (keys usable after face unlock) is sound, but two effects are undocumented or contradicted: (a) the journal and public-data keys are now usable with no screen lock at all, though the decision record says a lock screen is required before any key is used (the vault itself still refuses); (b) on a phone unlocked once since boot and then locked, code running as the app can strip the device layer and attempt an offline PBKDF2 brute force that skips the attempt counter. Several comments still describe the old behaviour.

Fix: add the lock-screen gate to the journal and public-data keys; correct the `KDoc` and `vault-crypto.md` and record the locked-phone exposure in the threat model.

BTS-15 — Default task affinity (Low)

Where: `app/src/main/AndroidManifest.xml` — `MainActivity` is exported, `singleTask`, with the default affinity (no `taskAffinity` attribute in source or merged manifest).

Issue: On older Android versions a malicious app can declare the same affinity and present a fake "enter your wallet password / recovery phrase" screen when the user opens the signer (StrandHogg-style). Plausible; depends on OS version.

Fix: `android:taskAffinity=""` on `MainActivity`.

BTS-16 — Backup verification can be passed by trial and error (Low)

Where: `WalletSetupViewModel.kt:419–431`; `core/model/.../Backup.kt:89–120`.

Issue: A wrong answer only clears the placements; blanks and chips stay the same, so at most 23 (4 blanks) or 119 (5 blanks) orderings exist, with no limit. The check then proves nothing about the user's written backup (matches iOS).

Fix: draw new blanks after a wrong answer, or return to "Review Phrase" after two or three failures.

BTS-17 — One Recovery Card of a 12-word phrase leaves ~40 bits (Low, design)

Where: `Backup.kt:25–28`; `RecoveryCards.kt:106–114`.

Issue: The 2-of-3 split is correct (verified), but one stolen card of a 12-word phrase leaves four words: 44 bits less 4 checksum bits, within reach of an offline GPU search. The app warns but still offers the cards.

Fix: offer Recovery Cards only for 18 words or more, or require an explicit acknowledgement for 12 and 15.

BTS-18 — Cancelled match in the keys-locked branch (Low, liveness)

Where: `DefaultSignerStore.prepare` (~792–801): the keys-locked `reload()` runs before the `try` whose cancellation handler sets `Interrupted`.

Issue: If the caller is cancelled there, the review stays on "Finding the matching wallet..." with no retry. Rare now that keys no longer carry the unlocked-device flag.

Fix: move that block inside the `try`.

BTS-19 — Partial overlays undetectable on Android 8.0–9 (Low, platform limit)

Where: `core/designsystem/.../security/Hardening.kt` — the partial-obscurity flag exists from API 29 and overlay hiding from API 31; `minSdk` is 26.

Issue: On API 26–28 an overlay covering only the amount or recipient cannot be detected. The password is still required.

Fix: raise `minSdk` to 29, or warn on `API < 29` about apps allowed to draw over others.

5. Informational notes

- **Engine host:** no CPU-time interrupt in QuickJS (a pathological valid PSBT could hang the engine thread; availability only); `GetStringUTFChars/GetByteArrayElements` return values not checked for NULL (JVM out-of-memory only).
- **Play signing:** the Play flavor is currently signed with the APK release key; configure a separate upload key before enrolling Play App Signing.
- **Attempt throttle** uses the wall clock: someone holding the unlocked phone can move the date to skip delays (documented). Optional: also bind to boot count and elapsed realtime.
- **Reveal vs delete:** revealing the phrase needs only the wallet password; deleting also asks for App Lock biometrics when enabled. Consider the same step before reveal.
- **Input methods:** the BIP39 passphrase and wallet passwords use the system keyboard (no personalised learning, password type); only the phrase uses the in-app keyboard.
- **Immutable copies:** the canonical phrase, revealed words and the engine's mnemonic response exist briefly as Strings/byte arrays that cannot or are not zeroed (inherent to JVM/Compose; zeroing the mnemonic response bytes is a cheap improvement).
- **Import draft** survives backgrounding (passwords are cleared; matches iOS).
- **Rename of a wallet whose public file fails its check** drops the duplicate-detection claim; later Repair flags both duplicates and signing refuses them.
- **FakeSignerStore** is compiled into the main source set (not bound; dead code).
- **Clipboard** expiry of copied addresses cannot run in the background on Android 10+ (public data, marked sensitive).
- **Dependencies:** `npm audit` reports 6 advisories (3 low, 3 moderate: elliptic GHSA-848j-6mx2-7j84, uuid GHSA-w5hq-g745-h8pq, via the Keystone reference packages); `npm audit --omit=dev` reports none, and none of these packages is in the shipped bundle (the bundle's `secp256k1` code is `@noble/curves`). No Maven/native CVE database scan was performed.

6. Residual risk

This is a software signer without a trusted display. A rooted OS, injected code, a malicious accessibility service or input method can exceed its assumptions. Memory wiping covers explicitly owned buffers only. Review cannot establish live contract behaviour, balances or final on-chain outcomes; risk acknowledgement is not a substitute for accurate summaries. JVM and Robolectric tests do not validate OEM Keystore behaviour, StrongBox, biometrics, overlays or capture on real devices. Before release, run the production build on representative API 26–28, 29–30, 31–34 and 35+ devices, including face/Smart Lock unlock, overlay consent, backgrounding during scanning and signing, and process death.

7. Remediation order

1. BTS-07 and BTS-08 in the iOS engine; re-sync with `tools/engine/sync-from-ios.sh`; add BTS-09 in the same change.
2. BTS-10: move the release key off the development machine; set `SIGNER_RELEASE_CERT_SHA256`; create the Play upload key.
3. BTS-11, BTS-12, BTS-15 and BTS-18 (small, contained Kotlin changes).
4. BTS-13, BTS-14, BTS-16, BTS-17, BTS-19 and the informational items as hardening.
5. Device qualification matrix, then an independent human audit.