

Peach Android

Read-only security, architecture, implementation-health, and release-integrity assessment of the complete pinned Android repository.

PUBLIC
DISTRIBUTION

v2.17.38
ASSESSED PRODUCT

562
FILES LEDGERED

Assessment control	Recorded value
Immutable baseline	Koh checkpoint 69f71a4ae3ec; source save 334d823aec8b
Assessment window	12 September 2026 UTC
Mode	Single-assessor, white-box, read-and-report
Report version	1.0
Classification	PUBLIC

Publication note

This report contains no personal attribution, workstation identifiers, live credentials, private service coordinates, or user data. It is written for public technical review.

1 Executive summary

Overall posture

The complete Peach Android 2.17.38 repository presents a mature source-level security posture. The assessment established no Critical, High, Medium, or Low source-backed vulnerability in the pinned baseline. Three Informational observations remain: an unbounded cross-client conformance runner, concentrated vault policy in one large repository class, and build/test migration warnings that reduce long-term signal quality.

Release posture

Source-level posture is conditionally favorable, not a business approval. Fresh execution completed 1,198 JVM tests with zero failures and the Android lint task succeeded. Manifest, native provenance, and release-helper gates passed. A later Koh record documents successful publication of both supported APK architectures with staged/live checksum equality. Device/OEM, hardware-backed key, production relay, Play distribution, and independent distributed-APK reproduction were not performed in this assessment.

Priority actions

- Make every cross-client scenario bounded and self-reporting before treating the matrix as a release gate.
- Reduce VaultRepository coupling incrementally when behavior next changes; avoid a broad rewrite.
- Retire compiler, lint, and Gradle migration warnings in isolated, behavior-preserving batches.
- Complete emulator/physical-device and external-service validation before making claims beyond source-level assurance.

0

CRITICAL/HIGH

0

MEDIUM/LOW

3

INFORMATIONAL

2 Scope, coverage, and limitations

Coverage result

All 562 files in the immutable snapshot were recorded once in the coverage ledger. Review depth was matched to material type: manual complete-file source review and call tracing for first-party code; test-quality and failure-path review for tests; manifest and policy review for resources; provenance and lock review for binaries and dependencies; and contextual reconciliation for documents and historical evidence.

Material class	Total	Disposition	Coverage
First-party source	227	Reviewed	100%
Tests and fixtures	180	Reviewed	100%
Resources and packaged assets	31	Reviewed	100%
Build and configuration	29	Reviewed	100%
Documents and contextual assets	95	Reviewed	100%

Material limitations

- No emulator or physical device was available; Android instrumented tests and StrongBox/Trusted Execution Environment behavior were not freshly exercised.
- No production relay administration, Play Console, staged rollout, signing ceremony, or license-service access was provided.
- The canonical Rust Noise source and sibling browser implementation are outside this repository snapshot; checked-in native libraries were provenance and allow-list verified, not source-audited here.
- The cross-client runner did not produce a bounded result in this execution, so remaining matrix scenarios are inconclusive.
- One assessor performed review, validation, severity calibration, and editorial synthesis; there was no independent second reviewer.

Scope exclusions

No attacks against real users, production accounts, shared infrastructure, device manufacturers, Android platform internals, package registries, or unrelated third parties were attempted. No production behavior was modified.

3 System overview and threat model

Protected assets

- Master-password equivalents; vault, root, recovery, device-identity, signing, and session keys.
- Vault entries, passkeys, notes, deleted history, recovery material, and duress-state confidentiality.
- Device trust, manifests, revocation state, reset generations, pairing state, and locked synchronization envelopes.
- Origin associations, vault size and activity metadata, entitlement state, and update trust material.

Principal trust boundaries

Boundary	Primary controls traced	Residual uncertainty
Locked to unlocked	Argon2id, AES-256-GCM, authenticated slots, session epochs, key clearing	Process-memory and device-hardware behavior
Framework caller to Autofill/Credential Provider	Binding permissions, package/signature/origin policy, private selection activities, secure windows	OEM framework variance
Relay or peer to trusted device	Noise sessions, bounded framing, signed manifests, immutable key history, revocation and generation state	Production relay and full cross-client run
Untrusted file/network to app state	Size limits, structured parsing, authenticated encryption, URI and checksum/package/signature checks	Device file-provider and installer behavior
Source to shipped APK	Dependency locking/verification, native allow-list, manifest drift, semantic version and signer gates	Independent rebuild comparison

4 Methodology

The assessment followed the campaign's ten work packages through six phases: immutable baseline and inventory; source-derived threat modeling; complete static review; controlled local validation; skeptical candidate validation; and publication synthesis. Automated searches increased consistency but did not replace complete-file review.

Review lenses

- Cryptography and storage: parameters, randomness, nonce/AAD use, authentication-before-use, padding, persistence, migration, rollback, and key lifetime.
- Android platform: manifest components, exported state, permissions, intent routing, PendingIntent identity, clipboard, notifications, secure windows, backup, and Keystore policy.
- Autofill and passkeys: AssistStructure trust, caller package/signature/origin binding, field matching, user verification, and result handoff.
- Sync and device trust: framing bounds, protocol negotiation, Noise boundary, pairing authority, manifests, revocation, reset generations, replay, and locked spool custody.
- Supply and release: repositories, lock state, dependency verification, native provenance, ABI/package/version/certificate gates, and failure-closed release behavior.
- Implementation health: responsibility, placement, dependency direction, large units, broad catches, logs, warnings, dead concepts, and test determinism.

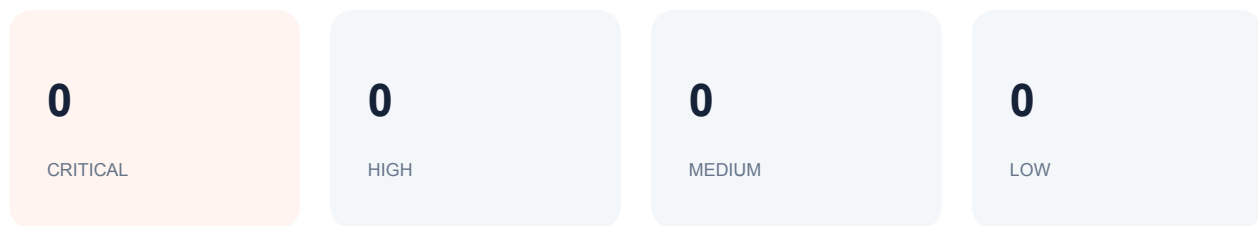
Severity and confidence

Severity follows realistic confidentiality, integrity, availability, privacy, and authority impact within the defined attacker model. Confidence is independent: High means directly reproduced or supported by unambiguous reachable code and verified preconditions; Moderate retains one material untested assumption; Low retains material uncertainty.

Evidence discipline

Evidence identifiers map claims to reproducible commands, source locations, or explicitly bounded analytical review. Negative conclusions apply only to the named paths and environment. Rejected candidates are not reported as vulnerabilities.

5 Results at a glance



ID	Severity / confidence	Title	Status
PB-26-201	Informational / High	Cross-client runner lacks bounded diagnostics	Open observation
PB-26-202	Informational / High	Vault policy concentrated in a 3,181-line class	Open observation
PB-26-203	Informational / High	Build and test migration warnings accumulate	Open observation

Validated absence within scope

No source-backed path was established by which a malicious co-installed application, unauthenticated exported component, compromised relay, stale or revoked peer, malformed ordinary import, or unsigned/mismatched update could obtain vault plaintext or silently expand root trust through the reviewed client paths. This is not a claim that vulnerabilities cannot exist, and it excludes the limitations stated in Section 2.

Work-package disposition

Work package	Disposition	Principal evidence
WP-01 Cryptography and key management	Complete static; device validation limited	PB-EV-002, PB-EV-003
WP-02 Sync, pairing, trust, revocation	Complete static; cross-client partial	PB-EV-002, PB-EV-005, PB-EV-007
WP-03 Android IPC and components	Complete static; instrumentation blocked	PB-EV-002, PB-EV-004
WP-04 Autofill and credential provider	Complete static; instrumentation blocked	PB-EV-002, PB-EV-003
WP-05 Persistence, import/export, recovery	Complete static and JVM tests	PB-EV-002, PB-EV-003
WP-06 Licensing and entitlement	Complete static and JVM tests	PB-EV-002, PB-EV-003
WP-07 Build, supply chain, release	Complete static and local helper gates	PB-EV-004 to PB-EV-006, PB-EV-009
WP-08 Regression and history	Complete through Koh history and delta review	PB-EV-001, PB-EV-002
WP-09 Code health	Complete with three observations	PB-EV-008
WP-10 Integrated posture	Complete within stated limits	PB-EV-002 to PB-EV-009

6.1 PB-26-201

Cross-client conformance runner stalls without a bounded diagnostic result

Attribute	Value
Severity	Informational
Confidence	High
Component	Cross-client assurance harness
Status	Open observation
Location	scripts/check-cross-client-sync.sh; CrossClientSyncProcessTest.kt

Technical description

The repository wrapper delegates directly to the sibling browser matrix. In this execution it produced neither progress output nor completion within 60 seconds. The process was stopped locally without modifying repository state.

Consequence

This does not demonstrate a protocol defect. It prevents the campaign from distinguishing slow setup, a blocked scenario, or a protocol hang and leaves non-completed matrix rows without fresh evidence.

Recommendation

Add a deadline and start/finish marker for each scenario, emit the active scenario on timeout, and produce a final machine-readable pass/fail/timeout table. Treat missing rows as gate failure.

Validation criteria

A later run should terminate within the declared campaign budget and assign exactly one disposition to every matrix row.

6.2 PB-26-202

Vault security policy is concentrated in one 3,181-line repository class

Attribute	Value
Severity	Informational
Confidence	High
Component	Vault architecture
Status	Open observation
Location	VaultRepository.kt

Technical description

VaultRepository owns encrypted persistence, session state, recovery, duress slots, Codex restore, key rotation, synchronization merge, migration, and rollback behavior. The current test suite exercises these paths extensively, but changes frequently cross unrelated invariants in one file.

Consequence

No exploitable bug was established. The concentration raises review and regression cost around the application's highest-value state transitions and makes ownership harder to reason about.

Recommendation

Avoid a broad rewrite. At the next related change, extract one cohesive transaction owner at a time, beginning with Codex restore/rollback and key rotation/rollback, while retaining the repository as the public facade.

Validation criteria

Each extraction should preserve byte formats and atomicity, reduce the repository line count, and pass the existing failure-injection and race tests unchanged.

6.3 PB-26-203

Compiler, test, lint, and Gradle migration debt is accumulating

Attribute	Value
Severity	Informational
Confidence	High
Component	Build and test maintainability
Status	Open observation
Location	AutofillResponseBuilder.kt; SettingsScreen.kt; VaultListScreen.kt; coroutine tests; Gradle output

Technical description

Fresh compilation reports deprecated Autofill dataset APIs, deprecated Compose nested-scroll names, missing ExperimentalCoroutinesApi opt-ins, Kapt language fallback, and Gradle features incompatible with Gradle 9. Android lint succeeds with 90 warnings and 5 informational notices.

Consequence

The current build is green and no security boundary failure was demonstrated. Accumulated warnings reduce signal-to-noise, can hide newly introduced diagnostics, and increase the risk of rushed platform upgrades.

Recommendation

Resolve in isolated batches: capture Gradle 9 warning-mode output and remove owned incompatibilities; migrate Autofill APIs with instrumentation coverage; use current Compose names; add explicit coroutine opt-ins in tests.

Validation criteria

Fresh rerun builds should remove the targeted warning family without increasing lint counts or changing security-sensitive behavior.

7 Validated attack chains

No end-to-end exploit chain met the reporting threshold in the pinned client source. The following abuse cases were traced to their limiting controls; each conclusion remains bounded by the dynamic limitations.

Abuse case	Control chain traced	Disposition
Malicious application invokes a privileged component	Exported surface inventory; framework binding permissions; private selection activities; intent routing	No unauthorized vault operation established
Spoofed web origin obtains credentials	Caller package resolution; browser signature verification; HTTPS origin/RP matching; explicit selection and verification	No accepted spoof path established in reviewed policy
Compromised relay expands device trust	Noise authentication; invitation binding; signed manifests; known-key history; revocation and generation checks	Relay-visible authority insufficient in reviewed client
Revoked device returns through replay or stale time	Signed revocation history; manifest epoch/version; terminal identity handling; secure reset generation	No replay path to effective trust established
Malformed import or update becomes executable/trusted state	Size/schema bounds; formula treatment; authenticated payloads; HTTPS/checksum/package/version/certificate checks	Malformed state rejected before privileged use

Defense-in-depth observations

- Security-sensitive persistence uses synchronous commit where atomic decision points require an observable success result; lint warnings about apply are therefore not blindly actionable.
- Release checks independently bind semantic version, ABI, package identity, native inventory, and established signing certificate.
- Locked synchronization material remains encrypted and bounded; current session epochs prevent late asynchronous work from mutating a newer lock state.
- Backup exclusions enumerate security-sensitive preferences across legacy, cloud-backup, and device-transfer policies while the manifest also disables backup.

8 Verified controls and strengths

Control	Verification performed / result	Scope limitation
Vault cryptography	Traced Argon2id parameters, UTF-8 input, AES-256-GCM, fresh nonces, AAD domains, authentication-before-use, padding, and KATs; JVM vectors passed	No side-channel or process-memory forensics
Keystore convenience unlock	Traced StrongBox request/fallback, authentication requirements, invalidation, attestation-shaped checks, and fail-closed unsupported paths	No physical hardware matrix
Autofill and passkey boundaries	Traced framework permissions, package/signature/origin policy, RP matching, user verification, one-shot identities, secure activities, and regression tests	No OEM/browser instrumentation
Sync device authority	Traced Noise JNI calls, bounded frames, identity history, signed manifests, revocation, generation reset, adoption, and spool custody	External Rust source and full matrix unavailable
Import and export	Traced bounds, structured decoding, AEAD failure behavior, spreadsheet formula defense, atomic replacement, and synthetic production-format tests	No hostile document-provider instrumentation
Update and release integrity	Manifest, ABI, package, semantic version, checksum, certificate-lineage, dependency verification, native provenance, and negative helper gates passed	No independent reproducible build
Failure-path testing	1,198 JVM tests passed, including corruption, replay, race, rollback, downgrade, boundary, and cross-format cases	JVM coverage does not substitute for device tests

9 Code health and security maintainability

The codebase has broad regression coverage and explicit security contracts. Its primary maintainability risk is concentration rather than abstraction sprawl: a small number of large policy files carry many high-value invariants. The preferred response is incremental decomposition tied to real changes, not a speculative architecture campaign.

Priority	Category	Location / consequence	Recommendation / effort
1	Assurance harness	Cross-client runner can stall without identifying the active scenario	Per-scenario timeout and result table / Small
2	Responsibility concentration	VaultRepository: 3,181 lines across multiple security transactions	Extract cohesive transactions only when next changed / Medium
3	Upgrade readiness	Deprecated Autofill/Compose APIs, coroutine opt-ins, Kapt fallback, Gradle 9 warnings	Resolve one warning family per verified change / Medium
4	UI concentration	SettingsCardBodies: 1,873 lines; changes are visually coupled	Split by existing settings domain when next modified / Medium
5	Lint signal	90 warnings include deliberate commit calls and policy-sensitive battery guidance	Document intentional warnings; eliminate actionable noise / Small

Dependencies

Dependency versions are locked and Gradle verification metadata pins artifact hashes and signer trust. Direct security-sensitive libraries include Bouncy Castle 1.84, OkHttp 4.12.0, zstd-jni 1.5.6-3, ZXing 3.5.3, and AndroidX security-crypto 1.1.0-alpha06. This assessment traced reachable use and repository controls but did not make a time-sensitive claim that all transitive dependencies are vulnerability-free.

Placement and boundaries

Domain-specific packages generally reflect ownership: vault, sync, passkeys, credentials, autofill, updates, and UI. The small shared package holds domain normalization and clipboard behavior used across surfaces; no new wrong-way framework dependency into pure cryptographic policy was identified. The principal placement candidate is functional decomposition inside VaultRepository rather than moving files between top-level packages.

10 Residual risk and release posture

Residual risk	Current evidence	Required next evidence
Device-specific Keystore and biometric behavior	Policy and JVM/static tests	StrongBox, TEE, fallback, invalidation, enrollment-change, and lockout matrix on supported API/OEM devices
Framework IPC/Autofill variance	Manifest and source policy; instrumented tests present	Run the committed instrumented suite across representative Android/browser combinations
Cross-client protocol completion	Source and JVM coverage; runner inconclusive	Bounded full scenario matrix with per-row evidence
External Rust Noise implementation	Packaged native allow-list and provenance	Source audit at pinned Cargo revision plus reproducible native build comparison
Production services and distribution	Local release gates; later publication record	Independent live endpoint, artifact, package, ABI, checksum, and signer verification

Posture statement

Within the pinned 2.17.38 client source and the tested threat model, no release-blocking source vulnerability was established. The source is suitable to advance through release governance provided the owner accepts the explicitly untested device, external-service, native-source, and cross-client assurance areas. This assessment does not approve release, accept risk, or guarantee absence of defects.

Conditions of validity

- The conclusion applies only to Koh checkpoint 69f71a4ae3ec, source save 334d823aec8b, and version 2.17.38.
- Any change to cryptography, trust, exported components, persistence, import/export, update checks, signing, dependencies, or native libraries requires targeted reassessment.
- A materially different merged manifest, certificate lineage, dependency graph, or distributed artifact falls outside this conclusion.

Conclusion

Peach Android 2.17.38 demonstrates a strong, layered client-side design for a high-value local-first password manager. The most important security properties are represented in code by explicit cryptographic domains, fail-closed parsing and state transitions, bounded synchronization messages, signed device authority, caller/origin controls, and redundant release integrity checks. Fresh JVM and lint execution supports those source conclusions.

The assessment found no validated exploitable vulnerability in the pinned repository. That result is necessarily narrower than saying the product is secure: hardware behavior, instrumented Android framework paths, the external Rust source, production services, and complete cross-client execution remain outside fresh proof. The three open observations should be treated as assurance and maintainability work, not as demonstrated user compromise.

Final assessment

Conditionally favorable source-level posture, with no validated Critical, High, Medium, or Low finding and three Informational observations.

Appendix A Environment and tools

Item	Recorded value
Repository state	Koh dev checkpoint 69f71a4ae3ec; clean at baseline
Product	Peach Android 2.17.38; minSdk 26; target/compile SDK 35
Build runtime	OpenJDK 17 for verification; Gradle 8.14.3; Kotlin 2.0.21
Host class	ARM64 macOS assessment environment; no unique host identifier retained
Static methods	Complete-file reading, call tracing, manifest/resource review, history delta review, targeted pattern sweeps, duplicate convergence scans
Dynamic methods	Fresh JVM tests and lint; repository helper gates; bounded cross-client attempt
Not available	Emulator, physical device, production relay/console access, independent distributed artifact rebuild

Appendix B Evidence index

Evidence	Description	Linked result
PB-EV-001	Pinned baseline and clean state	Baseline
PB-EV-002	Complete 562-file coverage ledger	WP-01 through WP-10
PB-EV-003	Fresh JVM and Android lint execution	Verification
PB-EV-004	Release manifest drift gate	WP-03 and WP-07
PB-EV-005	Native provenance allow-list gate	WP-02 and WP-07
PB-EV-006	Release helper negative-path gate	WP-07
PB-EV-007	Cross-client runner attempt	PB-26-201
PB-EV-008	Static control and hygiene sweeps	PB-26-202 and PB-26-203
PB-EV-009	Post-baseline distributed artifact verification	Release context

Terminology

Term	Meaning
AEAD	Authenticated encryption with associated data; encryption that also verifies integrity and context.
AAD	Associated authenticated data bound to ciphertext without being encrypted.
Argon2id	Memory-hard password-based key derivation function.
KAT	Known-answer test that pins exact cryptographic output for fixed input.
Noise	Framework for authenticated cryptographic handshakes; Peach uses native NK/KK patterns.
RP	Relying party in WebAuthn/passkey terminology.
StrongBox / TEE	Android hardware-backed security tiers used to protect cryptographic keys.
Trust manifest	Signed device-membership state used to define effective synchronization authority.

Appendix D Support package

The publication support directory preserves the complete coverage ledger, machine-readable finding register, evidence index, test record, review notes, report source, and deterministic report builder. These artifacts support reproducibility and do not expand the assessed baseline.

Report integrity

The final PDF SHA-256 is recorded in artifacts-sha256.txt after generation. All artifact paths are repository-relative and all content is public-safe.

END OF REPORT