[redacted] Mobile Application Black Box Assessment

[redacted] • May 2026 • Prepared by: Molly

CONTENTS

EXECUTIVE SUMMARY

We assessed the [redacted] Android application package [redacted] and the backend surfaces reachable from the mobile client. Testing combined static review of the APK and Hermes bundle with controlled dynamic validation from an Android 14 device used for the assessment and low-rate HTTP probes against the app-correlated backend and third-party integrations.

Four reportable issues were confirmed: unauthenticated provider detail exposure, authenticated support attachment uploads creating public S3 objects without a size policy, replay of an app-issued third-party chat SDK token to enumerate the chat directory, and an APK-exposed Google Maps key usable from an external client. No personal data, object identifiers, tokens, signed upload fields, or API keys were stored persistently at any point during the assessment; testing was conducted inside an isolated VM that was destroyed after the audit.

TARGETED ASSETS

`Android package [redacted]`

- **Version:** `[redacted], minSdk 24, targetSdk 36`
- **Checksum:** `sha256:[redacted]`
- **Notes:** Base APK extracted from Android 14/API 34 test device [redacted]. Split APK checksums were reviewed and are redacted in this copy.

`[redacted]`

- **Version:** `Backend API observed from Android bundle and runtime probes`
- **Notes:** Assessed unauthenticated and authenticated client-role routes including provider discovery/detail, profile, appointment, invoice, contact upload, notification, auth, refresh, documentation, and content endpoints.

`[redacted]-owned public S3 upload namespace`

- **Version:** `Presigned POST flow returned by [redacted]`
- **Notes:** Assessment was limited to one benign 51-byte PDF-like upload and unauthenticated HEAD checks. No high-volume storage testing was performed.

`[redacted]`

- **Version:** `Third-party chat SDK app id [redacted]`
- **Notes:** Only read-only replay checks using an app-issued chat token were performed. No messages, calls, group creation, or state-changing third-party chat SDK actions were attempted.

METHODOLOGY

OWASP MASVS | OWASP API Security | Mobile Static Review | Hermes Bundle Review | Authenticated API Review | Unauthenticated API Review | Third-Party Integration Review | ADB Runtime Validation | Manual Verification | Redacted Evidence Handling

The assessment used a mobile-first workflow. Static analysis identified package metadata, manifest exposure, deep-link routes, Hermes bundle constants, backend paths, third-party app identifiers, and client-side upload and chat flows. Dynamic validation used ADB device [redacted], authenticated client-account sessions where needed, unauthenticated controls, invalid-token controls, role-boundary checks, low-rate API probes, S3 HEAD/list checks, and redacted field-presence summaries. Findings were promoted only after candidate gates recorded a clear attacker model, entrypoint, trust boundary, observed/expected behavior, reproducibility steps, negative controls, impact, safety notes, and artifact references. Testing deliberately avoided destructive actions, high-volume uploads, contacting real users, or printing secrets into report artifacts.

TOOLS UTILIZED

JADX | apktool | Android Debug Bridge | Hermes bundle string and bytecode review | curl-based HTTP probes | jq | Custom validation scripts | S3 presigned POST and HEAD checks | Third-party chat SDK client REST API probes | Google Maps Geocoding API restriction probes | SQLite-backed assessment ledger

VULNERABILITIES

Unauthenticated provider detail API exposes provider financial, identity, and contact data

CVSS 8.7 HIGH

CWE-200

Description: The Android bundle exposes the public discovery flow using [redacted] followed by provider-detail requests to [redacted]. Dynamic validation against [redacted] showed that five sampled provider-detail requests returned HTTP 200 without an Authorization header.

The redacted evidence records sensitive field classes in those responses, including email, phone number, birth date, bank IBAN, business SIRET, exact address, KYC status, company photos, readiness flags, and uploaded diploma/file metadata.

Follow-up unauthenticated HEAD checks against referenced [redacted] URLs returned HTTP 200 for PDF/JPEG objects. A self-user-as-provider control returned 404, and unrelated protected routes such as appointments/pro returned 403, showing that the issue is specific to provider detail response shaping.

Impact: An unauthenticated internet attacker can enumerate public provider IDs from the public listing endpoint and then collect private provider profile, identity, financial, contact, and document metadata at scale. The exposed classes of data support privacy harm, targeted fraud, social engineering, financial-data exposure, and retrieval of uploaded credential documents.

Remediation: Define a strict public provider-detail response contract and enforce it server-side. Public provider views should return only fields required by the consumer profile experience. Financial data, identity verification state, exact private contact fields, uploaded credential metadata, and direct document URLs should require an authorized provider or administrative context. Add regression tests that call public provider-detail endpoints without Authorization and fail if private field classes are present.

Authenticated contact attachment pre-sign allows public S3 object creation without a size policy

CVSS 6.9 | MEDIUM

CWE-770

Description: The Android Hermes bundle contains a support-contact flow that posts file metadata to [redacted], uploads to the returned S3 presigned POST target, maps [redacted] into [redacted], and then posts [redacted].

Dynamic validation as a normal logged-in client account showed that [redacted] returned a presigned S3 POST before any contact ticket was created. Decoding the returned policy showed Content-Type and bucket/key conditions but no [redacted] condition.

A single benign 51-byte PDF-like object uploaded successfully with HTTP 204, and an unauthenticated HEAD to the returned public URL returned HTTP 200. The unauthenticated [redacted] control returned 403 and S3 bucket listing returned 403.

Impact: A low-privilege authenticated client can create publicly reachable objects in the [redacted] upload bucket without completing a support ticket. Because the signed POST policy does not enforce a server-side size bound, the grant appears to allow oversized attachment uploads until policy expiration, subject to S3 limits. This enables trusted-domain public file hosting and storage or bandwidth abuse from ordinary accounts.

Remediation: Bind contact attachment upload grants to a server-side contact draft or final contact record lifecycle. Include an explicit `content-length-range` condition in every S3 POST policy matching the product attachment limit. Prefer non-public object ACLs or delayed public availability until the contact submission succeeds. Expire unused grants quickly and run cleanup for orphaned objects. Add tests that decode returned policies and assert required size, content type, key prefix, and ownership constraints.

App-issued third-party chat SDK token can be replayed externally to enumerate chat user directory

CVSS 6.3 | MEDIUM

CWE-639

Description: The app login response included top-level [redacted], [redacted], [redacted], and [redacted] fields. Static correlation showed the Android app initializes a third-party chat SDK with app id [redacted] and calls [redacted], while normal in-app chat navigation is appointment-derived.

A curl-based external client replayed the app-issued [redacted] as the third-party chat SDK [redacted] header against the third-party chat SDK client REST API. Valid-token requests returned HTTP 200 for [redacted], [redacted], [redacted], and [redacted], while no-token and invalid-token controls returned HTTP 401 with expected authentication errors.

Redacted responses included uid, display name, avatar URL, role, availability status, lastActiveAt, and conversationId-shaped fields.

Impact: Any logged-in user who obtains their own chat token from the mobile client or API response can enumerate chat identities and presence metadata outside the intended app workflow. This can enable unsolicited contact attempts, user correlation, and social engineering. Message delivery, calls, group creation, and other state-changing third-party chat SDK actions were not attempted, so the confirmed impact is directory and metadata exposure.

Remediation: Do not expose a global third-party chat SDK directory to arbitrary logged-in clients. Configure third-party chat SDK permissions so tokens are scoped to relationships authorized by the application, such as appointment participants. Where product behavior requires lookup, proxy discovery through the backend and enforce appointment/user relationship checks before returning chat identities. Review third-party chat SDK roles, auth-token lifetime, and client API permissions, then add regression probes verifying that unrelated users cannot be listed or fetched.

APK-exposed Google Maps API key is usable from an unauthorized external client

CVSS 6.9 | MEDIUM

CWE-798

Description: Static review extracted multiple Google-style API keys from the Android bundle resources and Hermes strings.

A controlled external restriction probe sent one low-rate Google Maps Geocoding API request per discovered key from the harness environment, without Android package signature, referrer, or app-origin headers.

One redacted key index returned HTTP 200 with API status [redacted] for a benign [redacted] request. Two other Google-style keys from the same app returned [redacted], demonstrating that the replay method distinguishes restricted from unrestricted keys.

Impact: An attacker who extracts the public Android app bundle can replay the affected Google Maps API key from arbitrary external clients to consume Geocoding API quota. Depending on Google Cloud quotas and billing configuration, this can generate cost, exhaust quota for legitimate users, or create noisy third-party API usage attributed to the [redacted] project.

Remediation: Rotate the exposed key and restrict any mobile-distributed Google Maps keys by Android package name and signing certificate fingerprint. Apply API allowlists so each key can call only the required Google Maps APIs, set conservative quotas and budget alerts, and monitor for non-mobile usage patterns. Keep separate keys for Android SDK usage and server-side or web usage.

WEAKNESSES AND SECURITY NOTES

Refresh token replay remains accepted

LOW

CWE-613

Description: Refresh-token validation rejected invalid tokens and access-token-as-refresh controls, but replaying a valid refresh token minted fresh token material. No token disclosure, fixation, or cross-account primitive was demonstrated, so this was retained as a hardening note rather than a reportable vulnerability.

Impact: If a refresh token is later exposed through another weakness, replay acceptance can extend the useful lifetime of stolen session material.

Remediation: Use rotating refresh tokens with server-side reuse detection, revoke token families on replay, and record regression tests that verify old refresh tokens cannot be reused after rotation.

Account recovery and referral responses disclose enumeration signals

LOW

CWE-203

Description: Password reset and verification responses distinguish existing from non-existing emails, and public referral-code validation returns a boolean. Invalid reset tokens were rejected and no reset-token leak, password-change primitive, privilege escalation, or session-boundary crossing was demonstrated.

Impact: Enumeration signals can help an attacker build user or referral-code lists for phishing, spam, or credential-stuffing campaigns, even when account takeover is not directly possible.

Remediation: Normalize recovery responses and timing where feasible. Rate-limit email and referral validation flows by account, IP, and device identity. Monitor repeated failed recovery and referral checks.

Invoice fake-ID path returns server error before a useful authorization decision

LOW

CWE-248

Description: Authenticated GET sweeps found expected own-account data and provider-role controls, while unauthenticated controls were protected. A fake invoice ID path returned HTTP 500 before a useful authorization result, but no valid invoice disclosure primitive was demonstrated.

Impact: Server errors on malformed or nonexistent object identifiers reduce robustness and may reveal implementation behavior. They can also obscure authorization regression testing because the request fails before reaching a clear allow/deny decision.

Remediation: Validate object identifiers before downstream processing and return consistent 400 or 404 responses. Add tests for malformed, nonexistent, cross-account, and unauthenticated invoice requests.

Exported Android components increase exposed surface without confirmed impact

LOW

CWE-926

Description: Controlled shell-context probes reached exported providers, services, receivers, custom-scheme routes, app links, and a Notifee receiver activity. The probes did not produce unauthorized reads, app-owned state mutation, token or PII leakage, durable crashes, sensitive route navigation, or notification control. The WebPage component accepts `route.params.url` internally, but static and dynamic probes did not find an exported linking path to it.

Impact: Unnecessary exported components and broad deep-link surfaces increase the amount of code reachable from other apps and make future regressions more likely, even when current probes do not cross a security boundary.

Remediation: Mark components non-exported unless external invocation is required, protect exported services and receivers with signature permissions where possible, keep deep-link allowlists explicit, and add component-level regression tests for unauthorized intents and content providers.

Provider upload endpoints correctly enforce role, but upload policy coverage should be kept under regression

LOW

CWE-862

Description: Unauthenticated and client-role authenticated controls could not obtain provider upload pre-sign URLs for [redacted], [redacted], or [redacted]; well-formed numeric-size metadata returned 403 and no storage upload was performed. This was rejected as a vulnerability, but it is closely related to the confirmed contact-upload issue.

Impact: The current provider upload authorization boundary behaved correctly in testing. Future changes to upload grants could reintroduce cross-role or oversized-object risks if not covered by automated checks.

Remediation: Keep provider upload role checks in automated regression tests and apply the same policy requirements recommended for contact uploads: ownership binding, explicit content-length-range, strict content type, short expiry, and orphan cleanup.

Runtime storage and logging checks did not expose secrets

LOW

CWE-532

Description: Authenticated runtime checks found [redacted] blocked because the package is not debuggable, [redacted] unavailable, [redacted], private [redacted] inaccessible to shell, and no reusable [redacted] bearer, refresh, chat tokens, credentials, or authorization headers in the reviewed logcat captures after redaction. Remaining matches were framework lifecycle, Firebase, and deep-link noise.

Impact: No reportable secret exposure was confirmed. Continued discipline is still important because mobile logging and storage changes can accidentally expose reusable session material.

Remediation: Keep release builds non-debuggable with backup disabled, avoid logging tokens or PII, and add release smoke tests that scan launch/auth logs and accessible storage for reusable credentials.

Unauthenticated API sweep found expected public content and protected private routes

LOW

CWE-863

Description: Low-rate unauthenticated sweeps of bundle-referenced content and API paths found public FAQ, tariff, motif, and provider-listing data, while protected account, appointment, notification, report, address, review, and provider-role routes returned 403 or 404. This note excludes the separately confirmed provider-detail data exposure.

Impact: The broad unauthenticated API surface did not show a general authentication bypass. The main risk is that intended-public endpoints need explicit response contracts so private fields do not leak through public flows.

Remediation: Maintain an allowlist of intentionally public routes and field-level response contracts for each route. Add unauthenticated regression tests for protected endpoints and public-response snapshots that fail on private field classes.

Firebase storage, S3 bucket listing, and backend documentation exposure were denied

LOW

CWE-538

Description: Firebase project and storage identifiers were present in the app, but unauthenticated Firebase Storage list checks returned 404. S3 bucket and public-prefix listing checks returned HTTP 403 AccessDenied. Common backend documentation paths returned 404 JSON responses and did not expose Swagger or OpenAPI documents.

Impact: No public listing or documentation disclosure was confirmed. These checks provide useful negative coverage for future reassessment.

Remediation: Continue denying unauthenticated bucket listing and public documentation routes in production. Monitor storage access logs and keep infrastructure exposure checks in release or deployment validation.

CONCLUSION

The assessment confirmed four concrete, reproducible security issues in the Android client ecosystem.

The highest-priority fix is the unauthenticated provider-detail exposure because it leaks sensitive provider financial, identity, contact, address, KYC, and document metadata without any session.

The contact attachment upload and third-party chat SDK directory issues should follow because they allow low-privilege authenticated users to create public bucket objects or enumerate chat identities outside the intended workflow. The unrestricted Google Maps key should be rotated and locked down to reduce third-party quota and billing exposure.

Several other tested surfaces behaved correctly or lacked a demonstrated boundary crossing, including provider upload role checks, JWT forgery controls, protected unauthenticated API routes, runtime storage, logging, Firebase/S3 listing, and backend documentation paths.

The recommended next step is to fix the four confirmed issues and convert the negative controls from this assessment into automated regression tests.