MARIO ROBERTO FORTUNATO

2026-09-11

MOBILE SECURITY // ANDROID

# PENETRATION TEST REPORT

## Android Mobile Application Security Assessment

CLIENT / ENGAGEMENT

**[REDACTED]**
APPLICATION: [REDACTED]
PROJECT ID: 2026_02

PREPARED BY

**MARIO ROBERTO FORTUNATO**
dev[at]mariorobertofortunato.com
https://mariorobertofortunato.com

CLASSIFICATION

**PUBLIC**
REDACTED VERSION

# 00. REPORT CONTENTS

# 01. ASSESSMENT INFORMATION

## Engagement Scope Summary

| | |
|---|---|
| **ENGAGEMENT TIMEFRAME** | 2026-09-01 - 2026-09-11 |
| **PROJECT ID** | 2026_02 |
| **REPORT DATE** | 2026-09-11 |
| **ASSESSMENT TYPE** | Android Mobile Application Penetration Test |
| **ENGAGEMENT STATUS** | FINAL |

## Android Target Details

| | |
|---|---|
| **APPLICATION NAME** | [REDACTED] |
| **PACKAGE NAME** | [REDACTED] |
| **VERSION / VERSION CODE** | [REDACTED] |
| **APK / AAB SOURCE** | PLAY STORE |
| **APK SHA-256** | [REDACTED] |
| **TEST DEVICE / OS** | [REDACTED] |

## Report Author / Assessor

| | |
|---|---|
| **NAME / COMPANY** | Mario Roberto Fortunato |
| **ROLE** | Mobile Application Penetration Tester |
| **CONTACT** | dev[at]mariorobertofortunato.com |

## Client Details

| | |
|---|---|
| **ORGANIZATION** | [REDACTED] |
| **PRIMARY CONTACT** | [REDACTED] |
| **CONTACT** | [REDACTED] |

# 02. ENGAGEMENT OVERVIEW

Mario Roberto Fortunato conducted an Android Mobile Application Penetration Test for [REDACTED]. The assessment was intended to identify and validate exploitable security weaknesses in the authorized application and its supporting mobile-facing interfaces, then provide clear remediation guidance.

## Service Description

Mobile application penetration testing simulates realistic attacker techniques against an authorized Android application. The assessment combines (limited) automated tooling with manual review and targeted exploitation so that scanner output is verified and contextualized rather than reported as raw findings.

## Assessment Objectives

- Identify vulnerabilities in the Android application package, configuration, storage, inter-process communication, cryptographic use, authentication flows, and runtime behavior.
- Evaluate network communications and mobile-facing API interactions that are reachable through the tested application within the approved scope.
- Validate exploitability and assess both exploitation likelihood and potential impact.
- Provide actionable remediation guidance and evidence that can be reproduced by the client.
- Support optional retesting after remediation to confirm closure of reported issues.

## Assessment Approach

Testing prioritized manual verification and evidence-driven reporting. Automated findings were included only after validation, and severity was assigned based on the actual tested context rather than scanner defaults.

# 03. PROCESS AND METHODOLOGY

| | | |
|---|---|---|
| Phase 01 | **Reconnaissance & Acquisition** | Confirm the authorized build, collect identifiers and hashes, map application components, and document the test environment. |
| Phase 02 | **Static Analysis** | Review the AndroidManifest.xml, resources, decompiled code, secrets, exported components, cryptographic patterns, storage behavior, third-party libraries, and obfuscation controls. |
| Phase 03 | **Dynamic Analysis** | Exercise the application on an authorized test device or emulator, inspect runtime behavior, local storage, logs, IPC, WebViews, root-resilience controls, and sensitive data handling. |
| Phase 04 | **Network & Backend API Analysis** | Observe and manipulate application traffic through an authorized interception setup, assess TLS behavior, authentication, authorization, session handling, and API input validation. |
| Phase 05 | **Exploration & Verification** | Manually reproduce suspected issues, remove false positives, validate prerequisites, and capture evidence sufficient to support impact and remediation. |
| Phase 06 | **Assessment Reporting** | Prioritize confirmed findings by severity, summarize systemic weaknesses, provide remediation guidance, and record evidence and references. |
| Phase 07 | **Optional Retest** | Re-evaluate remediated findings and update closure status, residual risk, and supporting evidence. |

# 04. SCOPING AND RULES OF ENGAGEMENT

## Authorized Scope

| | |
|---|---|
| **ANDROID APPLICATION** | [REDACTED] |
| **BACKEND / API SCOPE** | [REDACTED] |
| **TEST ACCOUNTS** | None provide.<br>Regular accounts have been used for testing |
| **TESTING WINDOW** | 2026-09-01 - 2026-09-11 |

## Constraints and Rules

- The penetration testing has been carried within the agreed testing window.
- No indications have been given regarding the use of emulators or specific devices.
  We relied on our regular work machines declared in 01.ASSESSMENT INFORMATION
- No prohibited actions have been indicated.
  Nonetheless we relied on common sense regarding the preservation of the day-to-day operations of the application and its business.
  No destructive testing or denial-of-service has been carried out.
  No real customer data has been exposed or even viewed.
- No emergency contact has been provided.
  We carried out all communication with the Founder through the normal channels of communication.

## Sensitive Assets and Processes

[REDACTED]

## Out of Scope

No specific indication regarding out of scope asset has been provided.
As a rule of thumb, the current penetration testing only investigated the package downloadable from Google Play Store, and the APIs exposed to and used by the application: no assessment has been carried out regarding the security of the backend infrastructure per se.

# 05. EXECUTIVE SUMMARY

Mario Roberto Fortunato performed an Android Mobile Application Penetration Test for [REDACTED]. Testing was designed to proactively identify security weaknesses, validate their practical severity, and provide remediation guidance.
Specifically, [REDACTED] commissioned this assessment to evaluate the security posture of their application.

Because [REDACTED] also involves [REDACTED], testing focused on answering questions regarding the confidentiality of such information: can an attacker read, corrupt, or inject a user's [REDACTED] data without consent? Are the data of the user safe?

Based on the confirmed findings documented in this report, the overall assessed risk is **HIGH** (Exploitation Likelihood: **MEDIUM** x Potential Impact **CRITICAL =** Overall Risk **HIGH**).

The **CRITICAL** impact evaluation is derived from [REDACTED].

The **MEDIUM** likelihood of exploitation has been assessed considering [REDACTED].

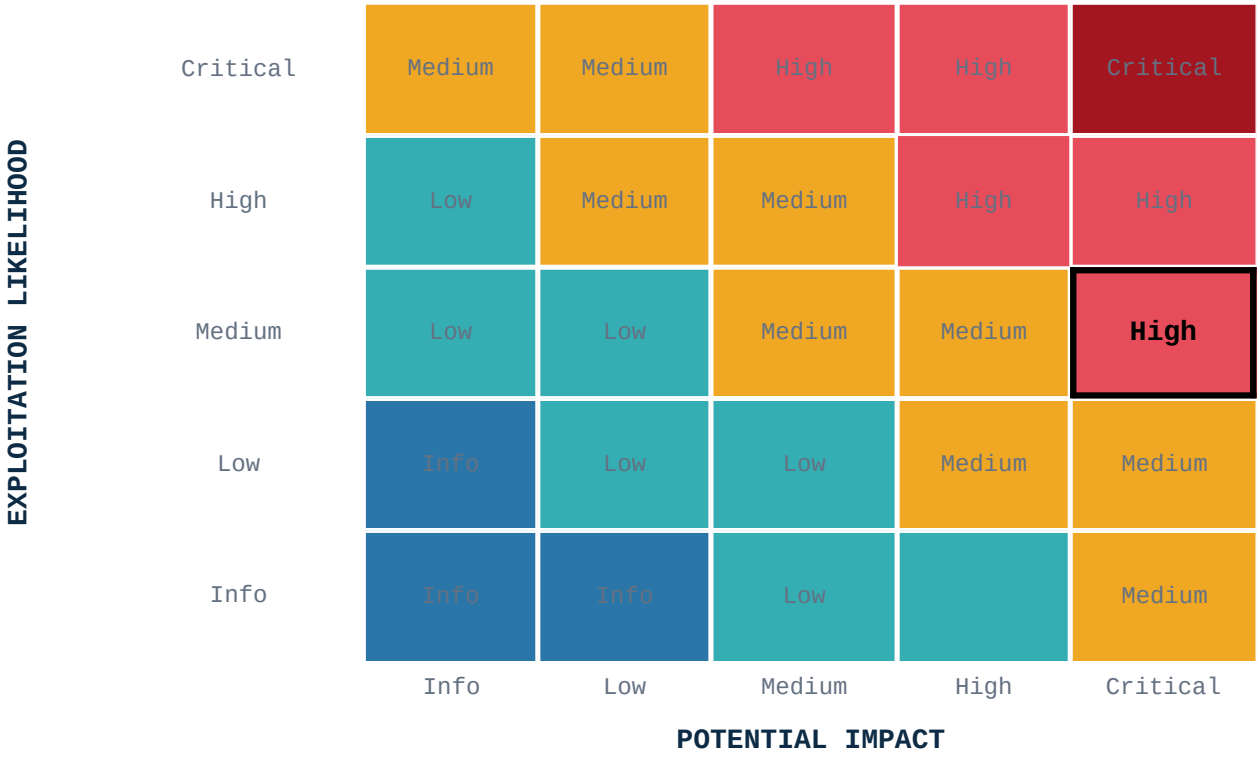
[REDACTED].

[REDACTED].

## Mobile Application Risk Rating

The Overall Risk Rating reflects the highest material unresolved risk identified during testing, with additional consideration given to the number of significant findings, their exploitability, affected business functions, and any compounding interactions between vulnerabilities. It is not calculated as an arithmetic average of individual finding scores.

**OVERALL RISK RATING: [HIGH]**




## Summary of Strengths

- The application is packaged with a decent amount of obfuscation, providing an effective first layer of security against reverse engineering.
- The overall concern for user security is clear throughout the app, which implements a relatively good level of cryptography for communication between users and for data persistence on the device.
- [REDACTED].

## Summary of Weaknesses

- [REDACTED]

## Strategic Recommendations

- [REDACTED]

## Finding Distribution

| CRITICAL | HIGH | MEDIUM | LOW | INFORMATIONAL |
|---|---|---|---|---|
| 1 | 3 | 1 | 0 | 1 |

# 06. VULNERABILITY OVERVIEW

## Vulnerability Risk Definition and Criteria

| | |
|---|---|
| **CRITICAL** | A severe issue that can enable catastrophic compromise, unrestricted sensitive-data exposure, or similarly extreme impact. Treat as an immediate remediation priority. |
| **HIGH** | A serious weakness with substantial security or business impact and realistic exploitation conditions. Remediate promptly. |
| **MEDIUM** | A meaningful vulnerability with moderate impact or additional prerequisites. Address after Critical and High items or sooner when business context increases risk. |
| **LOW** | A limited-impact weakness, hardening gap, or issue with significant exploitation constraints. Track and remediate proportionally. |
| **INFORMATIONAL** | An observation with little or no direct standalone security impact. Record for context, defense-in-depth, or future risk considerations. |

## Vulnerability Summary Table

| ID | FINDING | SEVERITY |
|---|---|---|
| **C-01** | Unrestricted Access to Session Data via Server Endpoints and Exploitation Chain Leveraging **H-01** and **H-02** | **Critical** |
| **H-01** | Hardcoded Secrets in Native Library | **High** |
| **H-02** | Plaintext Transmission of [REDACTED] via Third-Party Messaging Services | **High** |
| **H-03** | Exported [REDACTED] Allows Unauthorized Users to Join [REDACTED] Session | **High** |
| **M-01** | Insecure Network Security Configuration: Global Cleartext Traffic Permitted | **Medium** |
| **I-01** | Hardcoded Client Identifiers and API Keys (Public by Design) | **Informational** |

# 07. DETAILED FINDINGS

## Detailed Findings Summary

The following section provides the detailed technical analysis of each confirmed finding identified during the penetration testing assessment. Each finding is documented individually and includes the affected security condition, exploitation context, potential impact, recommended remediation, supporting evidence, and relevant security references or mappings.

Findings are assigned a unique identifier and risk rating to support consistent tracking throughout remediation and optional retesting activities. Where applicable, testing prerequisites, attack conditions, reproduction steps, and additional contextual information are included to ensure that each issue can be independently understood, reproduced, and addressed by the responsible technical teams.

Retest information may be appended to individual findings following remediation in order to document their updated status and any remaining residual risk.

# C-01 Unrestricted Access to Session Data via Server Endpoints and Exploitation Chain Leveraging H-01 and H-02

**Risk Rating: Critical**

Exploitation Likelihood: **HIGH** | Potential Impact: **CRITICAL**

## Description

An attacker who successfully exploits **H-01** and **H-02** can forge a valid *[REDACTED]* header and include it, together with a known *[REDACTED]*, in GET or POST requests to the server's [REDACTED] endpoint and its associated sub-endpoints.
The server does not verify whether the request originates from an authenticated, legitimately invited participant of the session: authorization is effectively delegated entirely to knowledge of the *[REDACTED]*, combined with a validly forged *[REDACTED]* header. As a result, possession of these two values is sufficient to impersonate a genuine session participant against the backend, regardless of whether the requester was ever actually invited to the session.

## Security Impact

A forged *[REDACTED]* header, combined with a *[REDACTED]*, grants access to a broad set of server endpoints. While some of these expose relatively limited information, others expose highly sensitive data belonging to [REDACTED]:

- [REDACTED]

[REDACTED]

## Remediation

- [REDACTED]

## Testing Process / Evidence

[REDACTED]

## References / Mapping

- [REDACTED]

# H-01 Harcoded Secrets in Native Library

**Risk Rating: High**

Exploitation Likelihood: **MEDIUM** | Potential Impact: **CRITICAL**

## Description

The application package includes a native library, *[REDACTED]*, which internally contains the *[REDACTED]* and *[REDACTED]* methods. By statically analyzing the compiled native code, it is possible to recover the hardcoded constants *[REDACTED]* and *[REDACTED]*. These constants are used across some sections of the application's business logic, most notably in crafting the *[REDACTED]* request header and in the encryption/decryption of confidential data in transit, such as [REDACTED].

Retrieving the value of these constants is relatively straightforward using tools such as Ghidra. While this is not within reach of a regular end user, it constitutes a feasible task for a moderately skilled malicious actor, meaning the effort to hide these secrets amounts to "obfuscation theater": it may deter casual inspection, but it does not withstand any meaningful pressure from a motivated attacker.

## Security Impact

[REDACTED]

## Remediation

- [REDACTED]

## Testing Process / Evidence

*[REDACTED]*

## References / Mapping

- CWE-798: Use of Hard-coded Credentials
- CWE-321: Use of Hard-coded Cryptographic Key
- OWASP MASVS-CRYPTO-1 / MASVS-STORAGE-1 (Mobile Application Security Verification Standard)
- OWASP MSTG-CRYPTO-1 / MSTG-STORAGE-14 (Mobile Security Testing Guide)

# H-02 Plaintext Transmission of [REDACTED] via Third-Party Messaging Services

**Risk Rating: High**

Exploitation Likelihood: **MEDIUM** | Potential Impact: **CRITICAL**

## Description

*[REDACTED]*

## Security Impact

[REDACTED]

## Remediation

[REDACTED]

## Testing Process / Evidence

## References / Mapping

- CWE-200: Exposure of Sensitive Information to an Unauthorized Actor
- CWE-522: Insufficiently Protected Credentials [REDACTED]
- OWASP MASVS-STORAGE-1 / MASVS-AUTH-3 (Mobile Application Security Verification Standard)
- OWASP MSTG-STORAGE-1 (data exposure via IPC/external channels)

# H-03 Exported [REDACTED] Allows Unauthorized Users to Join [REDACTED] Session

**Risk Rating: High**

Exploitation Likelihood: **HIGH** | Potential Impact: **HIGH**

## Description

Static analysis of the *AndroidManifest.xml* file revealed several exported Android Components. One of these is *[REDACTED]*, which is the component that [REDACTED].

[REDACTED]

## Security Impact

Because *[REDACTED]* is exported, no App Links verification, ownership check, or membership validation is performed before the activity is launched: the component trusts the *[REDACTED]* supplied in the intent data as sufficient proof of authorization to join the session. Consequently, any actor in possession of a valid *[REDACTED]* can join [REDACTED]

## Remediation

- [REDACTED]

## Testing Process / Evidence

[REDACTED]

## References / Mapping

- CWE-926: Improper Export of Android Application Components
- CWE-306: Missing Authentication for Critical Function
- CWE-863: Incorrect Authorization
- OWASP MASVS-PLATFORM-1 (Mobile Application Security Verification Standard) — IPC/component exposure
- OWASP MSTG-PLATFORM-4 — Testing App Permissions and IPC mechanisms

# M-01 Insecure Network Security Configuration: Global Cleartext Traffic Permitted

**Risk Rating: Medium**

Exploitation Likelihood: **MEDIUM** | Potential Impact: **HIGH**

## Description

The application's *network_security_config.xml* globally permits cleartext (unencrypted HTTP) traffic via *cleartextTrafficPermitted="true"* on the base-config element. Since base-config acts as the default policy for all domains the application communicates with unless overridden by a more specific domain-config, this setting disables a platform-level protection intended to prevent any component of the app, including third-party SDKs, WebViews, or future code changes from transmitting data over unencrypted HTTP.

Additionally, a domain-config entry explicitly whitelists cleartext traffic for the hardcoded IP address *[REDACTED]*, a private/local network address inconsistent with a production backend. This strongly suggests a development/debug endpoint left in the production build and reflects poor release hygiene, though it is not independently exploitable from a remote attacker's perspective due to its private address scope.

## Security Impact

By permitting cleartext traffic at the base-config level, the application removes the OS-enforced guarantee that all network communication is encrypted. This means:

Any current or future component (third-party libraries, ad/analytics SDKs, WebView content, misconfigured endpoints) that communicates over plain HTTP is not blocked by the platform, and any such traffic is vulnerable to interception and manipulation by an on-path attacker (e.g., on a shared or hostile Wi-Fi network, or via ARP spoofing on the local network).

This is a defense-in-depth failure: even without current evidence of sensitive data being sent over HTTP, the safety net that should prevent it is absent, and the risk surface grows with every dependency the app includes.

The exposed development endpoint (*[REDACTED]*) further indicates that non-production configuration was not stripped before release, and could aid an attacker who gains access to the same local network as the device in probing for a residual debug service.

## Remediation

- Set *cleartextTrafficPermitted="false"* on base-config, so cleartext is denied by default across the entire application.
- If any specific domain genuinely requires cleartext (e.g. a legacy internal service), scope it explicitly via its own domain-config entry rather than allowing it globally.
- Remove the domain-config entry for *[REDACTED]*, it appears to be a development/testing endpoint and should not be present in a production release.
- Review the build/release pipeline to ensure development-only network configuration is stripped or environment-specific (e.g. via build variants/flavors) before publishing to production.

## Testing Process / Evidence

[REDACTED]

## References / Mapping

- OWASP MASVS-NETWORK (MSTG-NETWORK-1)
- CWE-319 (Cleartext Transmission of Sensitive Information)

# I-01 Hardcoded Client Identifiers and API Keys (Public by Design)

**Risk Rating: Informational**

Exploitation Likelihood: **INFO** | Potential Impact: **LOW**

## Description

Static analysis of the decompiled application identified several hardcoded identifiers and API keys embedded in code and resources:

- *[REDACTED]*
- Firebase Crashlytics mapping file ID (*com.google.firebase.crashlytics.mapping_file_id*).
- Google API key (*google_api_key*).
- Google App ID (*google_app_id*).
- Google Crash Reporting API key (*google_crash_reporting_api_key*).
- Firebase Storage bucket name (*google_storage_bucket*).

These values are not secrets in the traditional sense because they are designed by Google and [REDACTED] to be embedded client-side:

- [REDACTED]
- Firebase's *google_api_key*, *google_app_id*, and related identifiers are standard contents of *google-services.json* and are expected to ship inside the application package; Google's security model for these relies on server-side restrictions, not on keeping the values confidential.

Regarding the Firebase Storage bucket specifically: although *google_storage_bucket* is configured, the firebase-storage module is not present in the build, confirmed via the *ComponentDiscoveryService* entries in the manifest (the mechanism Firebase uses to register active modules at runtime). No Storage functionality is reachable at runtime through this application, regardless of the bucket identifier being present in resources.

## Security Impact

None of these values are directly exploitable in isolation, as their exposure is expected by design. However, their effective security depends entirely on restrictions configured on the respective platform consoles, which could not be verified during this black-box, APK-only assessment:

- [REDACTED]
- If the Google API key lacks restrictions by Android package name and SHA-1 signing certificate fingerprint (configurable in Google Cloud Console), and if the same API key is enabled for other Google Cloud APIs beyond Firebase, it could potentially be extracted from the APK and reused to make calls against those APIs, leading to quota abuse or, in poorly configured projects, broader access than intended.
- Firebase Realtime Database/Firestore/Storage access (where applicable) is governed by Firebase Security Rules, not by the secrecy of *google_app_id/google_api_key*; misconfigured rules would be a separate, independently exploitable issue, but were not assessed as part of this finding.

This finding is therefore reported as **informational awareness**, not as a confirmed vulnerability, since verifying the actual restrictions requires access to the Google Cloud Console / Firebase Console / [REDACTED] account, which is outside the scope of an APK-only assessment.

## Remediation

- [REDACTED]
- Confirm on Google Cloud Console that *google_api_key* is restricted to the application's package name and release-signing SHA-1/SHA-256 fingerprint, and that it is not shared with unrelated, more sensitive Google Cloud APIs.
- Review Firebase Security Rules for any active Firebase services (Firestore, Realtime Database) to ensure access control does not rely on the confidentiality of client-side identifiers.
- No action is required regarding Storage specifically, as the module is not present in the current build; this should be re-assessed if *firebase-storage* is added in future releases.

## Testing Process / Evidence

[REDACTED]

## References / Mapping

N/A

# 08. RETEST SUMMARY

| ID | SEVERITY | STATUS | RETEST DATE | NOTES |
|---|---|---|---|---|
| C-01 | Critical | FIXED | 2026/09/10 | N/A |
| H-01 | High | FIXED | 2026/09/10 | N/A |
| H-02 | High | FIXED | 2026/09/10 | N/A |
| H-03 | High | FIXED | 2026/09/10 | N/A |
| M-01 | Medium | FIXED | 2026/09/10 | N/A |
| I-01 | Informational | FIXED | 2026/09/10 | N/A |

## Retest Conclusion

An informal retest has been carried out following the acceptance of the Confidential Report from part of the Client. The retest confirmed the successful remediation of the findings surfaced during the engagement.