ANDROID PAYMENT APPLICATION

Security Assessment Report

Static Analysis / Dependency Review / Manual Secure Code Review

DEMONSTRATION ASSESSMENT

ASSESSMENT ID	SEC-DEMO-ANDROID-2026-08
REPORT DATE	26 August 2026
TARGET	Android payment application (anonymous)
CLASSIFICATION	Client-facing sample / synthetic findings

REPORT SECTIONS

01 Executive verdict
02 Consolidated findings
03 Methodology & standards
04 MobSF / SonarQube / Lint
05 Dependency assessment
06 Manual secure code review
07 Remediation roadmap

All application names, code references, findings and evidence in this report are synthetic and provided for demonstration only.

Executive Verdict

Central security and quality oversight for the assessed Android payment application.

OVERALL POSTURE	
	High risk - release should be blocked until Priority 0 items are remediated. The synthetic assessment identified two critical and five high-severity weaknesses. The most material risks are retention of sensitive authentication data after authorization and a release-path TLS validation bypass. Both can directly undermine card-data confidentiality and trust in backend communication.

2 CRITICAL	5 HIGH	5 MEDIUM	3 LOW

Assessment scope. Review scenario assumes access to a signed release APK, application source code, Gradle dependency metadata, and a non-production API environment. The backend implementation, acquirer/processor infrastructure, payment kernel certification, terminal secure element, and production key ceremonies are outside the scope of this sample.

Key conclusion. The architecture is recoverable without a redesign, but the release pipeline needs stronger controls. Priority should be given to eliminating prohibited card-data retention, removing insecure TLS fallback behavior, replacing static shared secrets with per-device or short-lived credentials, and reducing externally reachable payment entry points.

Compliance note. The PCI observations in this report are security engineering guidance aligned to PCI DSS v4.0.1 concepts; this document is not a PCI DSS, PCI Secure Software, PCI PTS, MPoC, or EMVCo certification or attestation.

Highest-risk items

ID	Severity	Finding	Primary impact
F-01	Critical	Sensitive authentication data retained after authorization	PCI scope expansion; exposure of PAN/CVV from device storage or logs
F-02	Critical	Release TLS validation can fall back to trust-all behavior	Active network attacker can impersonate the API endpoint
F-03	High	Static backend credential embedded in the APK	Credential extraction and unauthorized API use
F-04	High	Exported payment entry activity trusts unsigned intent data	Local malicious app can invoke or manipulate transaction initiation
F-05	High	Release HTTP logging captures tokens and transaction fields	Sensitive data propagation into logcat / crash collection

Consolidated High-Risk Findings

Priority 0 and Priority 1 items aggregated across automated scanners and human review.

ID	Severity	Category	Primary detection	Summary
F-01	Critical	Data storage / PCI	Manual + SonarQube	Room retry cache and error logs retain full PAN and CVV after authorization.
F-02	Critical	Network security	MobSF + Manual	Custom X509TrustManager / HostnameVerifier accepts any server certificate on fallback path.
F-03	High	Secrets / authentication	MobSF + SonarQube	Long-lived API client secret is present in BuildConfig and used as backend authorization.
F-04	High	IPC / exported component	MobSF + Lint + Manual	Exported payment entry activity consumes caller-controlled amount/order fields without trust validation.
F-05	High	Logging / privacy	SonarQube + Manual	BODY-level HTTP logging remains active in release and logs bearer tokens and transaction payloads.
F-06	High	Session storage	MobSF + Manual	Refresh token stored in plaintext SharedPreferences and not excluded from device transfer rules.
F-07	High	WebView / platform	MobSF + Manual	Remote receipt/support WebView combines JavaScript bridge with caller-controlled navigation.

Priority 0: cross-tool validation

F-01 Sensitive data retention. Manual inspection establishes the payment-data context, while SonarQube detects the risky logging path and MobSF highlights unprotected local storage patterns. Because CVV is Sensitive Authentication Data, post-authorization storage is prohibited even when encrypted; the correct design is non-retention, not merely stronger encryption.

F-02 TLS validation bypass. MobSF flags the custom trust manager and permissive hostname verification implementation. Manual path tracing confirms the fallback client is reachable in the release flavor after specific handshake exceptions, converting a code smell into an exploitable network weakness.

F-03 Embedded credential. Both string extraction and source analysis recover the same long-lived secret. A mobile APK cannot safely hold a shared server secret; obfuscation may slow extraction but does not change the trust model.

RELEASE GATE	No production release until F-01 and F-02 are fixed and independently re-tested. F-03 to F-07 should be remediated in the same security sprint before live financial data is processed.

Methodology, Tools & Security Standards

Multi-tool analysis is used to reduce false positives and to verify exploitability in payment-specific context.

Assessment workflow

Step	Phase	Focus
1	Build & APK triage	Package metadata, signing, manifest, resources, embedded strings, release flags
2	Automated SAST	MobSF, SonarQube and Android Lint findings normalized and deduplicated
3	Dependency analysis	Gradle tree and OWASP Dependency-Check reviewed for known vulnerable components
4	Manual code review	Payment data lifecycle, authentication, IPC, WebView, cryptography, logging and error paths
5	Controlled runtime checks	ADB/logcat, local storage inspection, deep-link/IPC invocation, proxy-assisted TLS validation
6	Risk scoring & remediation	Business impact, exploitability, payment-data sensitivity and cross-tool confidence

Tools used in the demonstration

Tool	Use in assessment
MobSF	APK static analysis, manifest review, exported components, network-security configuration, hardcoded strings and risky APIs.
SonarQube	Java/Kotlin static analysis, hardcoded secrets, sensitive logging, reliability issues, code smells and maintainability risks.
Android Lint	Manifest/API misuse, component configuration, SDK compatibility and Android-specific correctness checks.
OWASP Dependency-Check	Software composition analysis against known CVEs in direct and transitive JVM/Android dependencies.
JADX / apktool / aapt2 / apksigner	Independent validation of decompilability, resources, manifest, signing metadata and release build configuration.
ADB / logcat + test device	Runtime evidence for logs, local storage, screenshots, activity invocation and lifecycle behavior.
Burp Suite / intercepting proxy	Controlled test-environment verification of TLS behavior and request contents; backend exploitation is out of scope.

Reference standards

Framework	How it is used
OWASP MASVS v2.1.0 / MASTG v2.0.0	Primary mobile security control baseline across storage, crypto, authentication, network, platform, code, resilience and privacy.
OWASP MASWE v1.0.0	Stable weakness taxonomy released 17 Aug 2026; used for finding-level mobile weakness IDs and traceability to MASVS/MASTG.
PCI DSS v4.0.1	Payment-data protection reference where cardholder data or Sensitive Authentication Data is in scope; not a certification claim.
CWE + CVSS v3.1	Weakness classification and severity support; final rating also considers payment-specific business impact.
Android Security Best Practices	Platform guidance for TLS, network security configuration, secure IPC, storage, WebView and UI protections.

Risk Domain Breakdown & MobSF Analysis

The static APK review identifies attack-surface exposure before source-level context is applied.

Domain	Findings	Risk signal
Storage & privacy	4	Critical - prohibited/over-retained payment data and tokens
Network	2	Critical - TLS validation bypass plus verbose request logging
Platform / IPC / WebView	4	High - exported entry points and untrusted WebView navigation
Resilience / build	2	Medium - weak reverse-engineering resistance
Reliability / dependencies	3	Medium - known CVEs and blocking calls in transaction paths

MobSF security analysis - representative results

Finding	Level	Static signal	Manual disposition
Custom TrustManager / permissive HostnameVerifier	High	Unsafe certificate validation pattern	Confirmed reachable in release fallback -> escalated to Critical (F-02)
Hardcoded credential-like string	High	Secret in BuildConfig/constants	Confirmed long-lived shared API credential (F-03)
Exported transaction activity	Warning	android:exported="true" with intent filters	Confirmed business-sensitive action is caller-controlled (F-04)
JavaScript-enabled WebView + bridge	Warning	addJavascriptInterface and remote URL loading	Confirmed unsafe navigation/bridge combination (F-07)
Unencrypted preferences	Warning	Auth material stored in SharedPreferences	Confirmed refresh token in plaintext (F-06)
Missing release hardening	Warning	Low obfuscation / readable classes	Accepted as resilience finding, not direct compromise (F-11)

Representative static evidence

```
<application android:networkSecurityConfig="@xml/network_security_config" ... /> class FallbackTrustManager : X509TrustManager { override fun checkServerTrusted(chain: Array<X509Certificate>, authType: String) { /* no-op */ } } hostnameVerifier = HostnameVerifier { _, _ -> true }
```

Interpretation. A permissive trust implementation is not automatically exploitable merely because it exists. In this demo, manual control-flow review confirms that the fallback client is instantiated by the release build following specific certificate errors, so the issue is treated as a genuine production-path vulnerability rather than a dead-code warning.

SonarQube Static Analysis Report

Source-level security, reliability and maintainability observations.

Summary dashboard

Metric	Demo count	Assessment
Security-related issues	8	Report severity after manual triage: 2 Critical, 4 High, 2 Medium
Security hotspots	37	Manual review required; 9 relevant to payment/security paths
Bugs	26	Reliability findings: blocking calls, ignored exceptions, lifecycle leaks
Code smells	214	Concentrated in transaction orchestration and legacy utilities
Automated test coverage	18%	Insufficient for security-sensitive refactoring

Tool context: SonarQube Server 2026.1 LTA - Standard Experience. Raw SonarQube issue types/severities are retained in the underlying scanner output. For this client-facing report, scanner results are normalized into the report severity model (Critical / High / Medium / Low / Info) after manual review; these labels are therefore assessment severities, not SonarQube-native severity fields.

Security analysis

HIGH

F-03 - Static backend credential embedded in APK

Mapping: MASVS-STORAGE-1; MASWE-0004; CWE-798 (authentication impact)

Impact. Any user or attacker who obtains the APK can recover the credential with standard decompilation or string extraction. If the backend treats this value as a trusted client secret, it enables unauthorized API access outside the app.

Synthetic evidence. A constant named `PAYMENT_API_SECRET` is copied into a release BuildConfig field and appended to an Authorization header by `ApiAuthInterceptor.kt`. The same value is visible in the decompiled APK.

Recommended remediation. Remove shared static credentials from the mobile trust model. Prefer user/session authentication plus per-device asymmetric keys generated in Android Keystore, hardware-backed where available, and short-lived server tokens. Rotate the exposed credential after rollout.

HIGH

F-05 - Sensitive request/response logging in release

Mapping: MASVS-STORAGE-2 / MASVS-PRIVACY; MASWE-0005

Impact. Bearer tokens and transaction attributes can leak into logcat, support captures or third-party crash telemetry. Android restricts log access for normal apps, but rooted, privileged, debug/support and diagnostic contexts remain relevant on POS fleets.

Synthetic evidence. The release flavor initializes OkHttp `HttpLoggingInterceptor.Level.BODY`. Manual test output includes Authorization headers, merchant identifiers and full JSON request bodies.

Recommended remediation. Disable BODY logging in release; redact Authorization, PAN-like fields and identifiers in all diagnostic layers. Add CI assertions that production artifacts use NONE/BASIC with explicit redaction.

Reliability notes

Blocking transaction calls. Several synchronous `Future.get()` calls have no timeout. A stalled provider callback can pin a worker indefinitely and can propagate into UI timeouts or duplicate retry behavior. Use bounded timeouts and cancellation tied to the transaction lifecycle.

Ignored interruption. Two worker loops catch `InterruptedException` and continue. Preserve the interrupted flag or exit the task so cancellation remains reliable during payment shutdown/reversal paths.

Dependency Security Assessment

Software composition analysis with exploitability context; CVE presence alone is not treated as proof of impact.

Executive dependency summary

Component	Observed version	Issue	Reachability / risk	Action
org.apache.commons:commons-compress	1.21	CVE-2024-25710 / CVE-2024-26308	Medium. Legacy diagnostic import exposes DUMP/Pack200 parsers to operator-supplied local support bundles; malformed inputs can hang parsing or exhaust resources.	Upgrade to 1.26.0+; disable unused archive handlers and bound parsing resources.
OkHttp logging-interceptor	4.x	Configuration risk, not a library CVE	High due to BODY logging in release (F-05).	Retain library; change production logging policy.
Legacy crypto helper AAR	vendor supplied	No reliable SBOM / provenance metadata	Medium supply-chain visibility risk.	Request SBOM/hash/version provenance; pin artifact checksum.
AndroidX / Kotlin libraries	mixed	Several versions behind current stable	Low by itself; creates patching and compatibility debt.	Upgrade incrementally with regression tests.

MEDIUM

F-12 - Vulnerable archive library in diagnostic import path

Mapping: MASVS-CODE-3; MASWE-0044; CWE-835 / CWE-770

Impact. A malicious or corrupted support bundle can trigger denial-of-service conditions during decompression. The risk is bounded because archive import is restricted to an operator-only diagnostic feature and does not process remote content automatically.

Synthetic evidence. Dependency tree resolves `commons-compress:1.21`. The legacy diagnostic import service exposes Commons Compress DUMP and Pack200 handlers for operator-supplied support bundles from shared/removable storage. CVE-2024-25710 affects 1.3-1.25.0; CVE-2024-26308 affects 1.21-<1.26.0.

Recommended remediation. Upgrade to 1.26.0 or newer. Disable unsupported archive formats and enforce input-size, memory, execution-time and cancellation limits around archive parsing; keep parsing off transaction-critical threads.

SCA NOTE

Dependency scanners can overstate risk when a vulnerable code path is unreachable. Every high-impact CVE should be paired with call-path or runtime evidence before being treated as a release blocker.

Supply-chain controls recommended

- Generate and retain an SBOM for every release artifact; include AAR/JAR hashes for vendor SDKs.
- Enable Gradle dependency verification or an equivalent checksum/signature policy.
- Pin repository origins and remove ad-hoc mirrors from production builds.
- Add a CI gate for critical/high CVEs with an explicit suppression workflow that records reachability evidence.

Android Lint & Platform Hardening

Android-specific correctness and attack-surface checks, separated from generic code-quality noise.

Summary dashboard

Category	Demo count	Security relevance
Manifest / exported components	6	High - two externally reachable flows require manual authorization review
API / SDK compatibility	14	Medium - legacy target behavior and deprecated APIs increase maintenance risk
PendingIntent / receiver usage	5	Medium - one mutable PendingIntent affects transaction notification flow
UI / privacy	7	Medium - sensitive screen can be captured; several fields allow clipboard/autofill
Resource / accessibility / localization	93	Low security relevance; tracked as quality debt

MEDIUM

F-08 - Mutable PendingIntent carries transaction context

Mapping: MASVS-PLATFORM-1; MASWE-0032

Impact. If another component gains access to the PendingIntent token, mutable extras can be altered before the receipt action is delivered. This is an integrity risk, not a direct card-data compromise.

Synthetic evidence. A receipt notification uses `PendingIntent.FLAG_UPDATE_CURRENT | FLAG_MUTABLE` and includes `transactionId` and `receiptMode` extras.

Recommended remediation. Use an explicit Intent and `FLAG_IMMUTABLE` unless mutation is strictly required. Resolve sensitive transaction state server-side or from trusted local storage instead of trusting mutable extras.

MEDIUM

F-10 - Sensitive card-entry UI can be captured

Mapping: MASVS-PLATFORM-3; MASWE-0038; PCI data-minimization context

Impact. PAN/CVV entered for a MOTO-style flow can appear in screenshots, screen recordings and recent-app previews. This expands exposure to support tooling, shoulder-surfing workflows and compromised screen-capture environments.

Synthetic evidence. The synthetic `ManualCardEntryActivity` does not set `FLAG_SECURE`; a test screenshot captured the active PAN/CVV fields.

Recommended remediation. Apply `WindowManager.LayoutParams.FLAG_SECURE` on sensitive activities. Also disable copy/paste/autofill where operationally appropriate and clear fields immediately after authorization.

LOW

F-13 - Legacy target SDK and deprecated platform behavior

Mapping: MASVS-CODE-1/3; MASWE-0042 / MASWE-0046

Impact. Older target behavior can delay adoption of newer Android security defaults and complicate predictable behavior across current POS firmware. It is a hardening and maintenance issue rather than a standalone exploit.

Synthetic evidence. Release configuration targets an older Android API while running on current terminal firmware; multiple deprecated permission and activity APIs remain in the transaction shell.

Recommended remediation. Raise targetSdk after regression testing, remove deprecated APIs in small batches, and validate payment SDK compatibility on the actual terminal firmware.

Manual Secure Code Review - Payment Data Lifecycle

Manual review provides the context scanners cannot infer: which values are PAN/SAD, when authorization completes, and what must be destroyed.

CRITICAL

F-01 - Sensitive authentication data retained after authorization

Mapping: MASVS-STORAGE-1/2; MASWE-0001 (unencrypted Room persistence); MASWE-0005 (logging); PCI DSS 3.3.1 / 3.3.1.2 (post-authorization SAD/CVV non-retention)

Impact. Full PAN and CVV are persisted in a Room retry entity and also included in one exception-log path. For a typical merchant/payment acceptance context, CVV is Sensitive Authentication Data and must not be stored after authorization - even if encrypted. Retaining full PAN also expands PCI scope and breach impact.

Synthetic evidence. The synthetic `PaymentRetryEntity` contains `pan`, `expiry` and `cvv`. A failed post-authorization receipt upload serializes the entity to a local retry table. A generic error logger can emit the same object as JSON.

Recommended remediation. Redesign retry state so it contains only a processor transaction reference, amount, masked PAN/last-four and non-sensitive receipt metadata. Never persist CVV post-authorization. Remove card data from object `toString()`/serialization paths, migrate/delete existing rows, and verify deletion on upgrade.

Illustrative vulnerable data model

```
@Entity data class PaymentRetryEntity( @PrimaryKey val transactionId: String, val pan: String, // full PAN - unnecessary for receipt retry val expiry: String, val cvv: String, // prohibited post-authorization SAD val processorReference: String, val payloadJson: String )
```

Safer model

```
@Entity data class ReceiptRetryEntity( @PrimaryKey val transactionId: String, val maskedPan: String, // e.g. ************1234 if needed val processorReference: String, val receiptPayload: String )
```

Important: Encrypting the original CVV field would not make the design acceptable after authorization. The remediation is to stop retaining the data and to prove that legacy rows, caches, logs and crash reports no longer contain it.

HIGH

F-06 - Refresh token stored without application-layer protection

Mapping: MASVS-STORAGE-1/2 / MASVS-AUTH; MASWE-0001 / MASWE-0006

Impact. A long-lived operator/session refresh token survives process death in plaintext preferences. Device compromise, backup/transfer behavior or privileged support tooling can expose it and prolong account access.

Synthetic evidence. `auth_prefs.xml` stores `refresh_token` directly. The manifest permits backup/transfer behavior without a dedicated exclusion rule for authentication data.

Recommended remediation. Prefer short-lived tokens and rotate refresh tokens. If a refresh credential must exist locally, protect it with Android Keystore-backed encryption and explicitly exclude authentication stores from backup/device-transfer rules. Invalidate on logout and device de-registration.

Manual Secure Code Review - Network, IPC & WebView

High-value Android attack paths: endpoint authentication, component exposure and trusted navigation.

CRITICAL

F-02 - TLS certificate and hostname validation bypass

Mapping: MASVS-NETWORK-1; MASWE-0027; CWE-295

Impact. An attacker controlling the network can present an arbitrary certificate and impersonate the payment backend whenever the fallback client is selected. This can expose credentials, transaction data and server responses to interception or modification.

Synthetic evidence. Release code catches specific TLS handshake failures and constructs a second OkHttpClient using an empty `checkServerTrusted()` plus `HostnameVerifier { _, _ -> true }`. A controlled proxy test confirms the client accepts an untrusted certificate.

Recommended remediation. Delete the fallback. Use the platform trust store and normal hostname verification. Keep test CAs in debug-only Network Security Configuration overrides. If the deployment requires stronger endpoint binding, use a carefully operated pinning/CT/mTLS strategy with rotation and recovery planning.

HIGH

F-04 - Exported payment entry activity trusts unsigned Intent extras

Mapping: MASVS-AUTH-1 / MASVS-PLATFORM-1; MASWE-0018 / MASWE-0029 / MASWE-0032; CWE-926

Impact. Any installed application can invoke the exported activity and supply amount, order ID and callback data. Even if the processor ultimately authorizes server-side, an attacker can create confusing transaction prompts, manipulate local pre-authorization state or abuse callback routing.

Synthetic evidence. `PaymentEntryActivity` is exported for a custom scheme. It reads `amount`, `orderId` and `returnUri` directly from extras without caller authorization or a server-signed transaction context.

Recommended remediation. Prefer verified App Links or a non-exported internal component. For intentional external integrations, require a signature-level permission or a cryptographically signed server-issued transaction request; validate amount/order state server-side and allowlist callback destinations.

HIGH

F-07 - Untrusted WebView navigation exposed to JavaScript bridge

Mapping: MASVS-PLATFORM-2; MASWE-0033 / MASWE-0035; CWE-749

Impact. A caller-controlled URL can be loaded into a JavaScript-enabled WebView that exposes a native bridge. If attacker-controlled content is loaded, JavaScript may invoke native methods intended only for trusted receipt/support pages.

Synthetic evidence. The synthetic `ReceiptWebActivity` enables JavaScript, registers `ReceiptBridge`, and accepts a `url` extra. Host validation is performed only with a substring check.

Recommended remediation. Remove the native bridge if not required. Otherwise load only a strict allowlist of HTTPS origins using parsed host equality, reject redirects to untrusted origins, disable unnecessary file/content access, and keep bridge methods minimal and non-sensitive.

NETWORK PRINCIPLE

The presence or absence of certificate pinning alone is not used as a vulnerability finding. The critical issue here is explicit bypass of normal certificate and hostname validation in a reachable release path.

Medium & Low Severity Findings

Defense-in-depth, privacy and reliability findings that should follow the release-blocking fixes.

MEDIUM

F-09 - Predictable client nonce / idempotency token

Mapping: MASVS-CRYPTO-1 / MASVS-AUTH; MASWE-0012

Impact. A predictable token weakens duplicate-request and replay defenses if the server relies on unpredictability rather than uniqueness plus authenticated server-side state.

Synthetic evidence. `java.util.Random` creates a six-digit request nonce used in the transaction header. Values repeat in a short runtime sample.

Recommended remediation. Use server-issued idempotency keys or generate a sufficiently large value with `SecureRandom`. Enforce uniqueness and replay windows on the backend; do not treat a client nonce as authorization.

MEDIUM

F-11 - Release build offers minimal reverse-engineering resistance

Mapping: MASVS-RESILIENCE-3; MASWE-0059

Impact. Readable class and method names materially lower the effort required to locate API flows, secrets and anti-tamper checks. Obfuscation is not a substitute for secure design, but it is reasonable defense-in-depth for a payment client.

Synthetic evidence. Release build has `minifyEnabled false`; source file names and meaningful internal class names remain visible after decompilation.

Recommended remediation. Enable R8 shrinking/obfuscation with tested keep rules for payment SDK reflection/JNI. Remove secrets regardless of obfuscation. Keep mapping files protected for crash analysis.

LOW

F-14 - Unbounded blocking waits in transaction orchestration

Mapping: MASVS-CODE; CWE-400

Impact. A provider or network callback that never completes can leave a transaction worker stuck, degrading availability and potentially causing unsafe user retries.

Synthetic evidence. Two flows call `Future.get()` without timeout; cancellation does not close the underlying network call.

Recommended remediation. Apply bounded timeouts, propagate cancellation to the SDK/network layer, and make retry/reversal state explicit and idempotent.

LOW

F-15 - Crash telemetry includes excessive identifiers

Mapping: MASVS-PRIVACY-2; MASWE-0067 / MASWE-0076

Impact. Crash events include terminal ID, merchant ID and cardholder name. These fields may not be prohibited card data by themselves, but they increase privacy exposure and data-retention obligations.

Synthetic evidence. The crash wrapper attaches the active `TransactionContext` object after redacting PAN but not operator/merchant/cardholder identifiers.

Recommended remediation. Adopt a telemetry allowlist, pseudonymize identifiers, define retention limits and confirm the processor/crash vendor receives no PAN/SAD or authentication tokens.

Additional quality observations

- Several payment coordinator classes exceed 1,500 lines and mix UI, device SDK, persistence and network responsibilities, increasing regression probability.
- Runtime permission checks are duplicated in multiple activities instead of a centralized capability layer.
- No automated test exercises reversal/idempotency behavior after process death or network timeout.
- Security regression tests are absent from CI; release security relies on manual review.

Critical Remediation Path

A staged plan focused on reducing immediate exploitability before broader technical debt.

Priority matrix

Priority	Target	Action	Exit criteria
P0 - Immediate	F-01	Remove PAN/CVV from retry entities, logs, crash payloads and caches. Add migration/deletion of existing data.	Instrumented tests prove no CVV/SAD persists after authorization; storage/log inspection clean.
P0 - Immediate	F-02	Delete trust-all TLS fallback and test production network configuration.	Untrusted CA/hostname rejected; only approved debug override can trust test CA.
P1 - Security sprint	F-03, F-04	Replace static shared secret; harden exported transaction entry and callback validation.	APK contains no reusable backend secret; unauthorized local invocation rejected.
P1 - Security sprint	F-05, F-06, F-07	Disable sensitive release logging, protect refresh tokens, constrain WebView origins/bridge.	Runtime evidence contains no secrets; backup rules validated; WebView rejects non-allowlisted origins.
P2 - Next sprint	F-08-F-12	Immutable PendingIntent, SecureRandom/server idempotency, FLAG_SECURE, R8, dependency upgrades.	Static scans clean or accepted with documented rationale; regression suite passes.
P3 - Hardening	F-13-F-15	SDK modernization, timeout/cancellation cleanup, telemetry minimization.	Compatibility matrix and privacy retention review completed.

Recommended retest suite

Test	Expected result
Storage sweep after successful / declined / timed-out transactions	No CVV, track data, PIN data, full PAN or bearer tokens in DB, files, cache, logcat or crash payloads.
TLS negative tests	Self-signed CA, wrong hostname and intercepting proxy certificates are rejected by release build.
APK reverse-engineering check	No long-lived server shared secret recoverable from resources, BuildConfig, native strings or assets.
IPC abuse tests	Untrusted package cannot start privileged transaction flow or control amount/order/callback without approved trust mechanism.
WebView origin/redirect tests	Bridge is unavailable to untrusted pages; non-allowlisted navigation and redirect chains are rejected.
Lifecycle / timeout tests	Network loss, process death and provider hangs resolve into bounded retry/reversal states without duplicate charge risk.

CI/CD GATES

Run Android Lint and SAST on every merge; run dependency/SBOM checks on every release build; archive signed APK hash and scan results; require security review for exported components, payment-data model changes, network trust changes and new third-party SDKs.

Suggested release decision: Block production use of live card data until P0 is closed. P1 should be completed before customer rollout. P2/P3 can be scheduled based on deployment threat model and terminal management controls.

Appendix - Severity Model, Assumptions & References

Supporting context for interpretation of this demonstration report.

Severity model

Level	Definition used in this report
Critical	Direct path to compromise of payment data, endpoint trust, authorization integrity or prohibited post-authorization SAD retention. Release blocker.
High	Material security weakness with realistic exploitation path or large blast radius; should be fixed before live production rollout.
Medium	Requires additional conditions, has bounded impact, or represents meaningful defense-in-depth / availability weakness.
Low	Limited direct security impact; hygiene, privacy minimization, maintenance or reliability issue with lower exploitability.

Assessment assumptions & limitations

- This is a **synthetic demonstration report**. No real client, application, source repository, cardholder data or production infrastructure was assessed.
- File names, class names, code snippets, counts, build settings and test evidence are intentionally fabricated but designed to represent realistic Android payment-app defects.
- Automated scanner output requires manual triage. The report intentionally distinguishes scanner signals from manually confirmed impact.
- PCI DSS references are included to demonstrate payment-data awareness; formal compliance requires the organization to determine scope and use the applicable validation process/assessor.
- Payment-kernel, PIN-entry, secure-element, acquirer host, terminal firmware and EMV certification testing are outside this mobile application security sample.

Reference basis

Reference	Version / purpose
OWASP Mobile Application Security Verification Standard (MASVS)	v2.1.0 - mobile security control baseline
OWASP Mobile Application Security Testing Guide (MASTG)	v2.0.0 - testing methodology and techniques
OWASP Mobile Application Security Weakness Enumeration (MASWE)	v1.0.0 (17 Aug 2026) - stable mobile weakness taxonomy and finding-level IDs
PCI Data Security Standard (PCI DSS)	v4.0.1 - protection of account data; SAD non-retention context
Android Developers - Security checklist & Network Security Configuration	Platform secure networking and application hardening guidance
NVD / Apache Commons Compress advisories	CVE-2024-25710 and CVE-2024-26308 dependency evidence
CWE / CVSS v3.1	Weakness classification and severity support

What a production engagement would add

A real assessment would replace the synthetic evidence with exact source paths and line numbers, reproducible proof-of-concept steps, APK hashes, device/OS matrix, scanner exports, raw dependency inventory, screenshots or packet evidence where appropriate, and a retest status for each remediation. The final report would also separate application findings from terminal/firmware, backend/API and payment-certification observations.

END OF REPORT

Demonstration security assessment prepared as a client-facing example of Android payment application review methodology. All findings are synthetic.